Getting Started with IBM WebSphere Development Studio Client for iSeries

Version 4.0 for Windows®

SC09-2625-06

Getting Started with IBM WebSphere Development Studio Client for iSeries

Version 4.0 for Windows®

SC09-2625-06

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 57.

Seventh Edition (May 2002)

This edition applies to Version 4.0 of WebSphere Development Studio Client and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC09-2625-05.

Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change or addition.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

IBM welcomes your comments. You can send your comments to:

IBM Canada Ltd. Laboratory
Information Development
B3/KB7/8200/MKM
8200 Warden Avenue
Markham, Ontario, Canada L6G 1C7

You can also send your comments electronically to IBM. See "How to send your comments" on page vi for a description of the methods.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

About this book

This guide introduces the IBM WebSphere Development Studio Client for iSeries (WDSC) product, and shows you how to develop and maintain applications using some of the features of the product.

Note: Important updates to this guide and to other product documentation may be available from the following Web address: http://www.ibm.com/software/ad/wdt400.

Who should use this guide

This guide is intended for iSeries application developers who want to maintain their current systems while exploiting the new technologies offered by the Internet and the World Wide Web. It is also intended for those interested in developing applications cooperatively using a programmable workstation, and wish to learn about the Java™ language support offered in WDSC. Those who are investigating how they can create client/server applications with graphical user interfaces (GUIs) will also benefit from reading this guide.

You should have a general knowledge of the iSeries system. You should also have experience in using applicable iSeries menus and displays, or control language (CL) commands.

Experience with a programmable workstation and either the Microsoft® Windows® 98, Windows NT®, or Windows 2000® operating system is also suggested.

WDSC information

Extensive online help is part of the WDSC product. This includes contextual user interface help, help on performing specific tasks, and online reference information.

Additional information is available at **ibm.com**/software/ad/wdt400/.

You can obtain information about other iSeries publications at http://publib.boulder.ibm.com/.

Related Web Sites

The following Web sites provide information on WDSC components:

- **ibm.com**/software/ad/wdt400/

Publications in PDF Format

All WDSC publications are included as part of the regular product help. In addition to the regular format, the product also contains the PDF version of some of these publications. You need the Adobe Acrobat Reader, Version 3.01 or later for Windows, to view the PDF files on your workstation. You can download a copy of the Reader from the Adobe Systems Web site at http://www.adobe.com.

Prerequisite and related information

See the Release Notes on the first product CD or the product DVD, under `x:\install\help\help`*`nls`*`\releaseNotes.htm` (where *nls* represents your language) for information on hardware and software requirements. See the readme.htm in the same directory for late-breaking information and known limitations. For more information on this product, visit **ibm.com**/software/ad/wdt400.

Use the iSeries Information Center as your starting point for looking up iSeries and AS/400e technical information. You can access the Information Center in two ways:

- From the following Web site:

  `http://www.ibm.com/eserver/iseries/infocenter`
- From CD-ROMs that ship with your Operating System/400 order:

  iSeries Information Center, SK3T-4091-00. This package also includes the PDF versions of iSeries manuals, *iSeries Information Center: Supplemental Manuals*, SK3T-4092-00, which replaces the Softcopy Library CD-ROM.

The iSeries Information Center contains advisors and important topics such as CL commands, system application programming interfaces (APIs), logical partitions, clustering, Java ™ , TCP/IP, Web serving, and secured networks. It also includes links to related IBM® Redbooks™ and Internet links to other IBM Web sites such as the Technical Studio and the IBM home page.

How to send your comments

Your feedback is important in helping us to provide the most accurate and high-quality information possible. IBM welcomes any comments about this book or any other iSeries documentation.

- If you prefer to send comments by mail, use the following address:

  IBM Canada Ltd. Laboratory
  Information Development
  D1/581/8200/MARKHAM
  8200 Warden Avenue
  Markham, Ontario, Canada L6G 1C7
- If you prefer to send comments electronically, use one of these e-mail addresses:
  - Comments on books:

    Internet: torrcf@ca.ibm.com
  - Comments on the iSeries Information Center:

    RCHINFOC@us.ibm.com

Be sure to include the following:

- The name of the book.
- The publication number of the book.
- The page number or topic to which your comment applies.

Conventions used in this guide

The following conventions distinguish different text elements:

- plain: Window titles, programming keywords, attributes, and method names.
- `monospace`: Programming examples, user input at the command line prompt or into an entry field, user output, and directory paths.

- **bold**: User interface controls (such as check boxes, buttons, icons, lists, menu choices, notebook tabs, folder names, and entry fields).
- *italics*: Programming variables, titles of documents, initial use of terms that are in the glossary, and emphasis.

The following conventions are used to abbreviate menu selections and object expansions within a user interface:

>
The right arrow when used within a menu shows a series of menu selections. For example, **File > New** is translated to mean: ″On the **File** menu, click **New**.″

The right arrow, when used within a tree view, denotes a series of folder (or object) expansions. For example, **Expand Management Zones > Sample Cell and Work Group Zone > Configurations** is translated to mean:

1. Expand **Management Zones**.
2. Expand **Sample Cell and Work Group Zone**.
3. Expand **Configurations**

+
The plus sign (+) denotes where a tree structure can be expanded to show more objects. To expand, click the plus sign (+) beside any object. If you double-click the object itself, a new tree structure is displayed with that object as the root of the tree.

-
The minus sign (-) denotes where a tree structure can be collapsed to remove its objects from view. To collapse, click the minus sign (-) beside any object.

left mouse button
The left mouse button is used for all actions in the application except for opening the pop-up menu of an object. Unless otherwise stated, the left mouse button is assumed.

right mouse button
The right mouse button opens the pop-up menu of an object. The pop-up menu contains a list of actions that can be performed on that object. The list of options varies depending on the type of object.

Chapter 1. Product Overview

You can quickly develop and deploy traditional and e-business applications on your iSeries system with IBM WebSphere Development Studio Client for iSeries. This powerful suite of tools represents the next generation of WebSphere development tools for iSeries and is the client component of the IBM WebSphere Development Studio product, which contains all of the host ILE compilers for RPG, C, C++, COBOL, and the Application Development ToolSet. This client component of the overall package is designed to help you accomplish three primary programming goals:

1. Develop and maintain iSeries business logic
2. Create Web front ends to iSeries business logic
3. Create GUI front ends to iSeries business logic

You can use this product to perform all of the following tasks from a Windows ® workstation.

Goal 1: Develop and maintain applications

- Connect to, explore, and perform programming tasks on remote systems
- Edit, compile, and debug applications, whether green-screen, Web-based, or combined
- Work on iSeries applications when disconnected from the iSeries system
- Develop iSeries business logic in the Java programming language
- Develop in a team-based environment with version control
- Create and edit DDS definitions visually, for traditional green-screen applications
- Deploy Web-enabled applications to an iSeries application server

Goal 2: Create Web front ends

- Generate Web front ends to existing iSeries applications
- Develop Java servlets and JavaServer Pages™ (JSPs) that access iSeries programs and data, whether your core skills are in Java or ILE programming
- Customize the appearance of a Web site's static HTML pages and JSPs
- Test Web applications on a local test environment

Goal 3: Create GUI front ends

- Develop graphical user interfaces to RPG programs using a visual design tool
- Produce Java applets and native Windows applications from the same source
- Hand-code Java-based graphical-based interfaces to ILE applications

Secondary Goals

- Develop XML resources
- Create Web Services
- Generate Web pages from local and remote relational databases

The next two sections, Tasks Overview and Tools Overview, describe in more detail the main tasks that IBM WebSphere Development Studio Client for iSeries addresses, and the product tools used for these tasks:

- Read the Tasks Overview if you are new to the product and want to understand how to choose the right tools to do your job.
- Read the Tools Overview if you have used a previous version of this product, or a similar product, and you want to understand what new tasks you can perform with each of the tools.

After you read one of the overview sections, you can try out one or more of the scenario-based samples described in subsequent chapters, to gain a more in-depth understanding of some of the tools you want to use.

Important Product Notes

1. If you have not yet installed the product, see the installation notes on the first product CD or the product DVD under x:\install\help\help*nls*\install.htm, where *nls* denotes the appropriate language.
2. IBM WebSphere Development Studio Client for iSeries takes you through the development and testing stages with the integrated server and test environment; to deploy and run your e-business applications live, you need to install and configure IBM WebSphere Application Server for iSeries and IBM HTTP Server for iSeries (powered by Apache) on the hosting iSeries system. Some configuration instructions and links to other sources of documentation are included in this Guide.
3. If you purchased IBM WebSphere Development Studio Client for iSeries as a workstation-only product, you need to ensure that WebSphere Development Studio is installed on any iSeries system where you do the host portion of your development work. WebSphere Development Studio is not required on the iSeries system you use for Web application deployment.

Tasks overview

The next section elaborates on the major goals and tasks IBM WebSphere Development Studio Client for iSeries addresses, by explaining, for each task, how the different product tools are used.

Goal 1: Develop and maintain applications

IBM WebSphere Development Studio Client for iSeries is designed to make the entire life cycle of developing and maintaining iSeries applications easier. Whether you work on traditional batch or green screen applications, or are responsible for making existing iSeries programs accessible as Web applications, your productivity is greatly increased as you work on the following types of tasks.

Edit, compile, and debug ILE applications

You can manage your development-cycle tasks in the Remote Systems Explorer. You can create and manage development projects on your iSeries system from your Windows-based workstation with this tool. With these tools, you can view iSeries libraries, files, and members; launch the host compilers, the workstation editor or debugger.

Your program editing tasks are simplified by the LPEX editor. The editor can access source files on either your workstation or your iSeries system directly. When a compilation results in errors, you can jump from the compiler messages to an editor containing the source positioned at the offending source statements so that you can correct them.

With the Distributed Debugger you can debug your program running on the iSeries system from a graphical user interface on your workstation. The tools helps you set breakpoints, monitor variables and storage locations, step through program instructions, and examine the call stack.

Develop iSeries business logic in the Java programming language

You can use Java development tools, provided with IBM WebSphere Development Studio Client for iSeries, to create Java applications that can access iSeries data. Java development tools are extended to provide iSeries support so that you can call native iSeries applications from your Java code.

Create and edit DDS definitions easily

You can create and update DDS definitions for display files, printer files, and physical files using CODE Designer when you are on or offline. This tool provides a graphical user interface with which you can quickly define your DDS files, and save them locally or upload them directly to your iSeries system. By creating your DDS screens visually you can improve their usability (because more of your time is spent on visual design, and less on getting the syntax right) and your programming productivity.

Goal 2: Create Web front ends

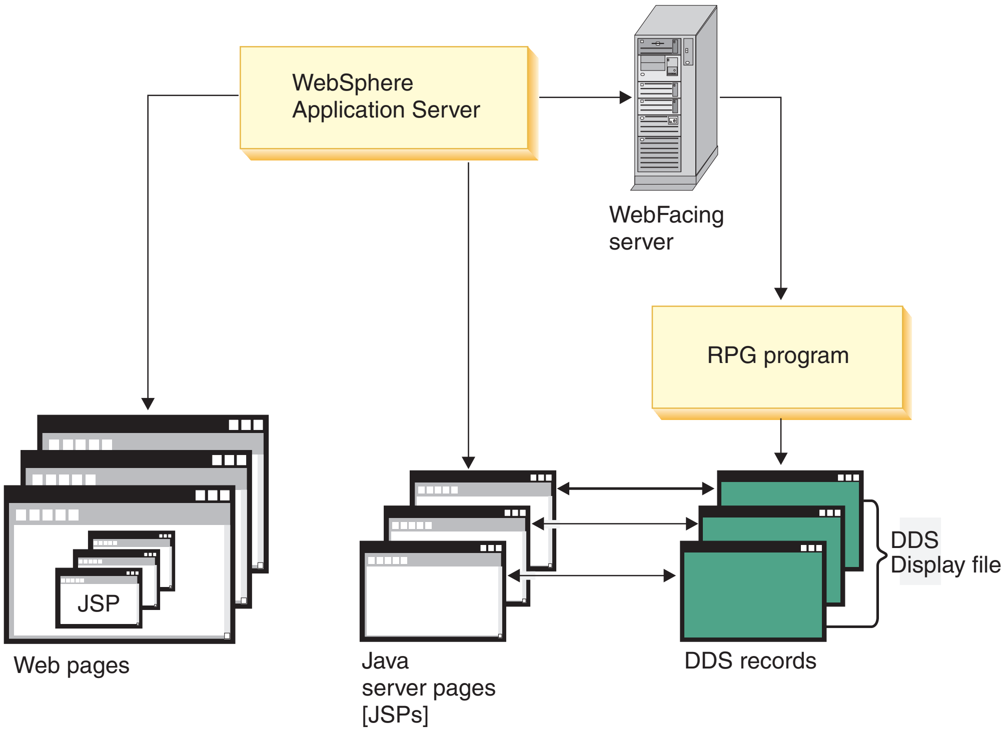
IBM WebSphere Development Studio Client for iSeries gives you a number of ways of making your iSeries applications and data accessible beyond the green-screen interface. You can generate Web-based look-alikes for your program's DDS screens, or create a new Web interface that connects directly to your program's input and output parameters. Which path you choose depends on the way your iSeries application is designed, and on trade-offs you need to make between rapid deployment and customized look-and-feel. With IBM WebSphere Development Studio Client for iSeries you can perform the following Web-enabling tasks for your iSeries applications.

Generate Web front ends to existing applications

You can use the WebFacing Tool to create Web front ends to iSeries applications that use DDS for their green-screen transactions. You choose a Web style for your green-screen application, generate a set of JSPs and JavaBeans™ that interact with the logic of your program, and then easily test your application in the unit test environment of the workbench. When you are ready to deploy your well-tested application, you can generate WAR and EAR files that can import the Web application into the WebSphere Application Server, based on J2EE standards.

WebFacing Tool is ideal for applications that you want to deploy broadly over a corporate intranet or the Internet, where rapid deployment takes precedence over customizing each page's look and feel. With the WebFacing Tool you can continue to deliver your application as a 5250 application, and use the same ILE programs to deliver the application through the Web. Note that there are some restrictions on which DDS keywords can be converted, generating sets of JSPs and Java Beans, described in the WebFacing documentation.

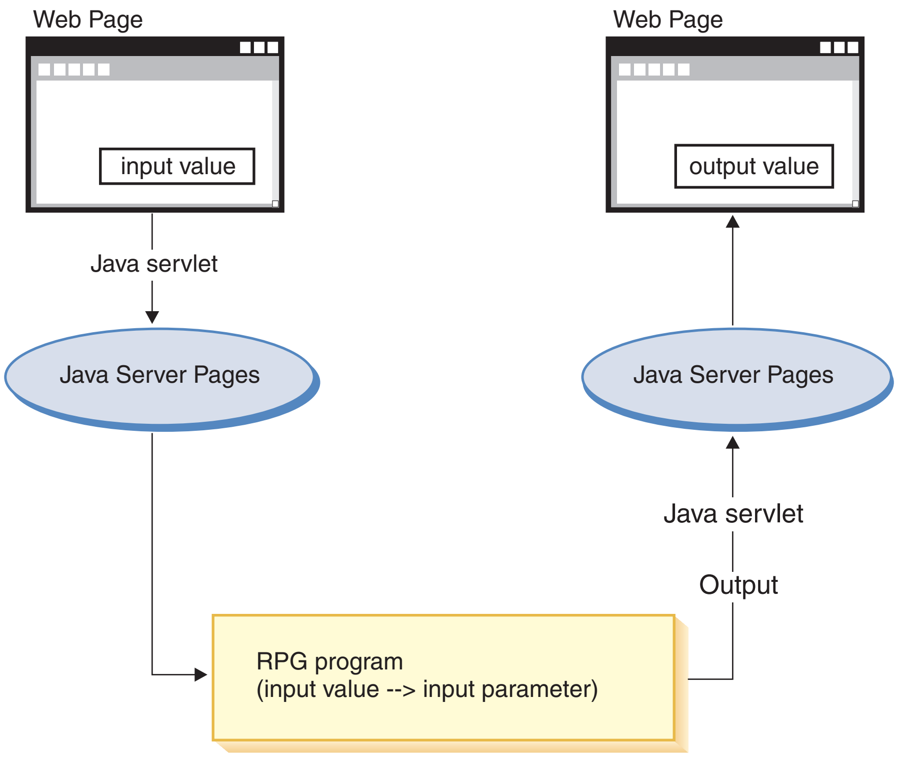
The following diagram illustrates the the WebFacing Tool's run-time process:



Use the WebFacing Tool for programs where the DDS user interface is interwoven with the business logic. If you architect programs with a set of distinct entry points for each element of business logic, and a user-interface component that simply generates appropriate calls to business logic entry points based on user inputs, you should consider using Web development tools to develop your own servlets and JSPs to access your code, described in the next task.

Develop Java servlets and JSPs to access iSeries programs and data

You can create interactive Web pages, and define how those pages interact with one or more ILE applications, using the Web Interaction wizard of Web development tools. This wizard generates Java servlets and JSPs for you that provide data to your ILE programs based on inputs the end-user enters in an HTML form, and that transform the outputs of your ILE program into Web-formatted output. Your ILE logic can be separated into different programs for each stage of input and output (known as a Web transaction), or can be a single service program with entry points to handle each Web interaction. The following diagram illustrates the Web development tools process:



You can also use Java development tools to develop customized Java servlets that run on the iSeries server and make use of iSeries data either directly, or through ILE applications. IBM Toolbox for Java extensions to Java development tools let you access iSeries data files and issue ILE program calls from the Java servlets. You develop the servlets using Java development tools, and use the Record I/O and Program Call classes from the toolbox to retrieve and update iSeries data. You can launch your servlets, or customize their output from the JSPs that you create in the Page Designer editor in Web development tools.

Customize the appearance of your Web site's static pages

You can easily customize your HTML and JSPs using the editing tools in Web development tools. These tools let you create and update input forms, change the appearance or placement of chunks of text, and add backgrounds and images to your pages. You can also add controls to handle input errors or to provide fancy touches like scrolling banners; the editors generate the appropriate JavaScript™ code for you. The iSeries-specific design-time controls let you create Web versions of your input and output pages with the same kinds of input validation, output formatting, subfile controls that native DDS screens provide. You do not need a detailed knowledge of HTML or JavaScript to accomplish these tasks.

Test and deploy your Web-enabled applications

You need an easy test mechanism for your Web-enabled applications. These include applications you have hand-coded on your workstation that use HTML, JSPs, Java servlets, and applications whose Web front-end you create using one of the code-generating wizards in Web development tools. You can easily run your program in the unit test environment of the workbench, quickly make changes, and re-test, rather than re-deploy your application every time you want to verify functions. When you are finished, you can package and deploy your web applications as Web archive (WAR) files and generated Enterprise Archive (EAR) files; entities that can import a Web application into the WebSphere Application

Server, based on the new J2EE standards. When you choose to deploy your work, Web development tools takes care of creating the necessary files and folders on the iSeries system, and also of creating or checking for the WebSphere Application Server services that your application needs. Web development tools also takes care of stopping and restarting the required WebSphere Application Server services after deployment.

Goal 3: Create GUI front ends

You do not have to deploy your iSeries applications on the Web in order to give them a more modern look and feel. You can produce graphical user interface (GUI) front ends to your iSeries applications with Java development tools and VisualAge RPG.

Develop graphical user interfaces to RPG programs

If you are already an experienced RPGIV programmer, you can create graphical user interfaces to RPG programs very quickly in VisualAge RPG, by selecting visual parts from a parts palette and dropping them on a representation of your program's user interface. You can alter the size and text of push buttons and check boxes, and associate user events, such as the pressing of a button, with VARPG code you develop. The VARPG code can include statements to access iSeries databases on a specific host system. The compiled program runs as an executable on a Windows system, or as an applet in a Java Virtual Machine in a Web browser, and accesses remote data on the iSeries system using TCP/IP. The only skills you need to develop these thick-client applications are the RPG skills you already have, and the ability to learn the straightforward user interface of the VisualAge RPG GUI Designer. The communications code is generated for you in the compiled program.

Produce applets and native graphical user interfaces from the same source

After you create a visual interface and associated VARPG logic on your workstation, you can deploy your application either as a native Windows application, or as a Java applet that can run in any Web browser with an appropriate Java Virtual Machine (JVM). This gives you extra flexibility in getting your application out to users. If you want to control access tightly, you can deploy the application as a Windows executable. If you want widespread access you can choose to place your applet (and associated files generated by VisualAge RPG) on a Web site, so that users with browser access to the Web site can run the applet in their browser and communicate with data on the iSeries system.

Hand-code Java-based graphical user interfaces to ILE applications

Use your own custom calls to Java GUI classes (e.g. Swing classes) to develop your graphical user interface, and then use classes provided by IBM Toolbox for Java or Java Beans provided by Java development tools to access your iSeries system. Although this option gives you great flexibility over your program's look and feel, it requires more work and skill than using VisualAge RPG, or than developing HTML-based interfaces to iSeries programs using Web development tools.

Tools overview

This section describes the main tools of IBM WebSphere Development Studio Client for iSeries, and the kinds of tasks you can perform with each tool. The tools are:

- “New — The workbench-based integrated development environment”
  - “New — iSeries development tools”
  - “New — Web development tools” on page 8
  - “New — Java development tools” on page 9
  - “New — Web services development tools” on page 9
  - “New — XML development tools” on page 10
  - “New — Server development tools” on page 10
  - “New — Database development tools” on page 11
  - “The WebFacing Tool” on page 11
- “VisualAge RPG” on page 11
- “CODE” on page 12
- “Distributed Debugger” on page 14

New — The workbench-based integrated development environment

IBM WebSphere Development Studio Client for iSeries uses the new WebSphere Studio Workbench (WSWB). The workbench platform offers you e-business tools in a comprehensive development environment that integrates all your various tools, teams, assets, and other workbench-based projects seamlessly. The platform provides the core frameworks and services upon which all plug-in extensions are created, the run-time in which plug-ins are loaded, integrated, and executed, plus a common user interface model for working with tools. Plug-in tools can program to the workbench portable Application Program Interface, yet run unchanged on any of the supported operating systems. Thus, the architecture discovers these plug-ins and integrates them all into the new base environment, providing a standard user navigation model for building integrated web and application development tooling projects.

You can customize your workbench desktop with the specific views or perspectives (task-oriented combinations of views) you need for everyday work, and still have access to other functions if you need them later. Because development resources for a project are stored in a single repository, the workbench is ideal for resource sharing and for providing consistent team support for shared projects.

New — iSeries development tools

iSeries development tools encompass the framework, user interface, editing, and file, command, and job subsystems of iSeries capability. In summary, it contains the following features:

Remote Systems Explorer

This workbench perspective supports views, editors, tools, and tool extensions, where you can develop applications while working with remote operating systems. The interface gives you access to files and resources, and the ability to monitor jobs and run commands. The Remote Systems Explorer is a programming environment, providing similar support as CODE Project Organizer but in the workbench. The Remote Systems Explorer also supports other system types, such as UNIX® or

Linux; it guides you through various connections to your remote systems; and it provides support for the sharing of work through a team perspective.

Universal file and command subsystems

The Remote Systems Explorer works with any remote system that supports Java. You can export, import, explore remote files, and execute remote commands for the iSeries Integrated File System, iSeries Native File System, UNIX, Linux, and Windows systems.

iSeries exploration and command execution

Use the workbench IDE to list libraries, objects, and source members to your local machine, and organize these elements into filters. You can create and share filters for rapid access to specific artifacts and commands. Filters can also be partitioned into pools to allow for easier organization of filters. Specific subsystems can handle Native File System (QSYS) files and CL commands, different than the generic file and command subsystems that involve only Integrated File System files and QSHELL commands. You can create your own actions and commands, or use the ones supplied by IBM for any remote object.

iSeries actions for edit, compile, run, and debug

The Remote Systems Explorer contains the LPEX editor, so you can edit source code directly in the workbench. The interface also provides right-click actions so that you can compile and run your programs plus debug them with the Distributed Debugger.

New — Web development tools

Web development tools give you the ability to create new e-business applications that use a Web-based front end to communicate with the business logic in an ILE language program residing on an iSeries system. You can create your Web input and output pages with the Page Designer, or generate input and output pages from the Web Interaction wizard templates. You can also add iSeries design-time controls to your pages, for example, Web equivalents of iSeries command keys, input fields that accept only particular types of data, or subfile and other iSeries output fields.

Web development tools are the workbench-based enhancement of WebSphere Studio for iSeries, giving you all the same features plus open standards, greater flexibility, and the ability to tie in existing applications with Web services wizards that let you write applications that are portable across platforms.

Web development tools include the following features:

- Integrated visual layout tool for JSP/HTML file creation, validation, editing, and debugging
- Compliance with Java 2 Enterprise Edition (J2EE) specifications and hierarchy, giving you access to enterprise systems, thereby helping you create web projects
- Maximum programming performance, with automatic web link updates, JavaScript media tools for image editing and animation, built-in Java refactoring tools, XML file creation and editing, transformation of XML using Visual XML, an integrated unit-test environment and a distributed, remote debugger
- Integrated support for the WebSphere Application Server (WAS)
- HTTP/FTP import, FTP export to a server, and Web Application Archive (WAR) file import and export
- a WebArt Designer tool that helps you create graphic titles, logos, buttons, and photo frames

- Link viewing, parsing, and management, so that Web development tools update links when resources move or are renamed
- a Servlet Wizard that can create servlets and related files
- Wizards that generate Web applications from database queries and beans
- Integration in the Web development tools unit test environment

The specific iSeries extensions of Web development tools include:

- a Web Interaction wizard that links the Design-Time Control (DTC) fields for a web page to parameters of your Program Call and defines the Program Call parameters without the need to deal directly with JavaServer Pages^{TM}, JavaScript code, or servlet code
- All existing DTCs available in the classic WebSphere Studio for iSeries, and a new table design-time control that helps you construct tables with multiple columns consisting of input fields that can be mapped onto Host program input and output parameters in the Web Interaction Wizard

New — Java development tools

Java development tools are the workbench-based enhancements of VisualAge for Java Professional edition, giving you similar features plus various wizards and editors that encompass Web Application development as well as Java development, letting you write, compile, test, debug, and edit programs written in the Java programming language. Java development tools include an integrated environment that supports the complete cycle of Java program development. It is the ideal tool for:

- Developing and compiling business logic in Java that runs on an iSeries system and accesses iSeries data
- Creating applications and Java GUIs that access existing iSeries data, business logic, and resources
- Deploying or exporting your Java class or source files to an iSeries server

The specific iSeries extensions of Java development tools include:

- export to and import from the Integrated File System
- a Program Call wizard that can create Java Beans and Program Call Markup Language (PCML) files to call your iSeries programs or service programs. The Java Beans can also be used to create Web Services using Web services development tools.
- The IBM Toolbox for Java
- the following pre-supplied beans from Java development tools:
  - Data File Utility bean — to extend the support of code to access one or more iSeries database files, and manipulate records in the files. You can create an interface similar to the Subfile with these Java beans.
  - Swing JFormatted bean — to let you create GUI fields with edit code, edit word formatting, and verification capabilities.
  - Object List bean — a redesigned set of Java Beans that let you access iSeries object names (for example, libraries, objects, files, members, records, or fields).
- special Java ″compile″ view for remotely compiling Java classes and ″run″ view for remotely running Java applications

New — Web services development tools

Web Services tackle the problem of inefficient distributed computing, and Web services development tools allow you to develop this service. When components in a system are tightly coupled, and based on database records and flat files, they are rigid and sensitive to change. System components need to be loosely coupled and dynamically bound to provide greater flexibility, scalability, less overhead costs, and hence, better business growth. Web Services are self-contained, modular applications that can be described, published, located, and invoked over the World Wide Web, employing 'just-in-time' integration of services. The architecture involves a relationship between Service providers, brokers, and requesters. More specifically, Service requesters submit a request to a Service broker, which finds the right service from a Service provider.

The Web Service wizard can create a Web Service, using a JavaBean generated by the iSeries ProgramCall wizard, to call one or more legacy programs or service program procedures on the iSeries. More specifically, the Web Service wizard works in the context of a Web Project and allows for creation, deployment, testing, generation of a proxy, and publication to a UDDI registry of Web Services based on Java Beans, input and output URLs, DB2® XML extender calls, DB2 Stored Procedures, and SQL queries.

New — XML development tools

The XML development tools environment includes tools for building DTDs, XML Schemas, XML, XSLT, and for mappings between XML and different back-end files such as an SQL query or relational table. The environment provides you with the following:

- XML Editor — a tool for creating and viewing XML files
- DTD editor — a tool for creating and viewing DTDs
- XML schema editor — a tool for creating, viewing and validating XML schemas
- XSL trace editor — a tool for XSL transformation scripts, and transformation rule highlighting
- XML to XML mapping editor — a tool that helps you map one or more source XML files to a single target XML file
- XML and SQL query — a wizard that helps you create an XML file from results of an SQL query
- RDB to XML mapping editor — a tool for defining the mapping among one or more relational tables and an XML file

New — Server development tools

Server development tools let you test your applications in locally or remotely installed run-time environments. You do so by creating a server project that represents your environment and can associate your Web projects with a server configuration. This tells your server tools how to configure the instance for that particular Web project.

Server development tools include the following components and features:

- a local copy of the WebSphere Application Server run-time environment so that you can test Web and Enterprise Application Archive projects
- a local copy of the TCP/IP Monitoring Server run-time environment, which forwards requests and responses, and monitors test activity
- support for the local Apache TomCat run-time environment so that you can test Web projects that contain servlets and JSPs

- support for Remote Agent Controller (installed on your remote machine) if you want to let WAS create a server project instance and configuration for you when you run your Web application (rather than having you creating the server project yourself)

  Note: If you use an external server instance, such as WAS on your Windows machine or remote iSeries machine, you need to deploy your project before running. To deploy, you can use Server development tools to create a remote file transfer instance to handle the details of how and where to copy files.

New — Database development tools

Database development tools support any local or remote database that has a Java Database Connectivity (JDBC) driver. Database development tools contain the following features:

- Relational Schema Center — views to enable database, table, view, index, and key creation
- Online and offline support — metadata generated as XMI
- SQL Query Builder — visually constructed SQL statements that perform SQL and XML mapping

The WebFacing Tool

With the WebFacing Tool, also available in the workbench, you can quickly convert your DDS display file source members so that the user interface of your iSeries programs can run in a browser.. When you convert your DDS display files, JSPs and JavaBeans are generated for you that substitute for the DDS code and make Web access possible. In the WebFacing wizard, you can select one or more DDS source members to convert, and select a Web look and feel from one of several predefined styles, or you can design your own Web style for use with your applications. The tool creates three JavaBeans and one JSP per record format; the JavaBeans hold the data for the record format, or control its appearance or other characteristics, and the JSP handles displaying the Web version of the screen, prompting for data, and handling input errors. The wizard generates an application home page to launch the Web-enabled version of your program.

When a user invokes a WebFaced application from the browser, the WebFacing server on the iSeries system starts the host program. The server intercepts all calls to READ, WRITE, and EXFMT operations to DSPFs, so that in many cases your program (*PGM) can run without modifications, and without even detecting that it is being accessed using WebFacing. You might need to make coding changes if your application uses DDS keywords that are not supported by WebFacing, or if you want to modify the DDS screens so that the conversion to Web format produces a more attractive or consistent result.

VisualAge RPG

With VisualAge RPG you can develop and maintain client/server applications in a visual development environment on the workstation. You leverage your existing RPG skills to create graphical user interface (GUI) applications that you can deploy as native Windows applications, or as Java applets that can run on any Java-capable Web browser. These applications can access iSeries data and other iSeries objects.

With VisualAge RPG you can create, edit, compile, and debug applications on your workstation. You can build an application from the top down. You start by focusing on the look and feel of the interface, and then you tie all the parts together with workstation RPG logic that you write in the VARPG language. You can reuse RPG logic and import display files (DSPF) from an existing application.

You can learn VisualAge RPG quickly thanks to its tightly integrated tools and its visual development environment. For example, you can quickly create text, buttons, and fields in your design window using a point-and-click action.

VisualAge® RPG includes the following features and tools:

- You can create interfaces in the GUI Designer by selecting visual parts and dropping them onto your design window, rather than by coding UI capabilities in source code. Select a part such as a push button or an entry field, drag it with the mouse, and drop it on the window you are designing. You can then select an event for the part from the part's popup menu, and use the editor to create the workstation RPG program logic behind the event. From the GUI Designer, you can also launch the editor, the compiler, and the debugger.
- The syntax checker helps you detect syntax errors quickly in your VARPG programs.
- The compiler performs compilations on the workstation, where your applications eventually run.
- The help and message compiler allows you to incorporate online help and messages into your workstation application.
- The editor allows you to add new editor functions or change existing ones. Language-sensitive editing and language-sensitive help are provided with the editor.
- For Windows 98 and Windows Me, you can use the VisualAge RPG debugger to debug your code: set breakpoints, monitor variables, registers, the call stack, and storage; debug VARPG or C++ applications; and customize the debugging session. For Windows NT and Windows 2000, the Distributed Debugger provides equivalent function.

CODE

CODE gives you a suite of utilities for creating source and DDS files, and managing your projects. CODE consists of CODE Designer, CODE Editor, and CODE Program Generator.

CODE Designer

CODE Designer removes the drudgery from managing your DDS files. You perform the following DDS editing tasks with CODE Designer:

- Create new DDS screens, printer files, and physical files
- Add text to DDS screens, simply by clicking and typing
- Insert new screen fields for input, output or both, from a popup menu
- Manipulate fields and text using drag-and-drop actions
- Change the attributes and properties of data fields and plain text
- Link DDS fields to iSeries database fields, by connecting to the database and selecting appropriate fields from a pull-down list
- View the hierarchical relationships among files, records, fields, help specifications, keys, and keywords in each selected DDS object
- Organize records into groups for a particular screen, report, or printer file

You can view the DDS source for each element when you add or update it. You can even edit the DDS code that CODE Designer generates for you. CODE Designer also saves you time by flagging errors before the DDS source is compiled on the iSeries system.

CODE Editor

CODE Editor provides many powerful editing features for file management:

- Seamless access to local and host files
- Member locking of OS/400 members and Application Development Manager parts that is similar to SEU (source entry utility)
- Concurrent multiple edit sessions as well as multiple views of the same document
- Navigation through procedures and subroutines in one or more files
- Multiple file search and a file compare utility.

Other features of CODE Editor that you might find particularly useful include the following:

- Language-sensitive editing that offers tokenizing, token highlighting, interactive syntax checking, and online language-sensitive help for C, C++, REXX, CL, Java, RPG, COBOL and DDS, as well as program verifiers for some of these languages
- Capability to view only specific types, such as C specs or lines with a specific string pattern
- Block move, insert, delete, and case switching actions
- Sequence numbers, which allow the use of SEU-style commands in the prefix (sequence number) area
- Command shell for running local or host commands with logged output

With CODE Editor you can develop not only source code, but custom programs that run in the editor to make you more productive. It provides these programming features:

- Ability to customize and program the editor using REXX or Java. You can also use the set of Java macros or “Lpexlets” that are provided with the product, to help you understand your code. (To use these macros, you must have the Java 2 Software Development Kit [J2SDK] from JavaSoft installed on your workstation. See the *Installation Notes* on the product CD or DVD for more information.)
- Powerful regular expression searching.
- Keystroke recording and playback.

CODE Program Generator

You can use CODE Program Generator on your workstation to:

- Specify compiler and linker options
- Submit requests to compile, bind, or build objects on the host or local workstation

You can also compile and link your C and C++ programs from a Windows command line:

- The **ixlc** command can compile source files residing on your Windows client, and can be used in conjunction with make files to control the compilation process.
- The **ixlclink** command links modules residing on an iSeries system to create bound programs.

You can also use **ixlc** as a host-based command, using the Qshell interface from a 5250 emulation window.

Distributed Debugger

The Distributed Debugger helps you debug code that is running on the iSeries system or on your Windows system, using a graphical user interface on your workstation. It supports the following tasks:

- Setting breakpoints on a source code line or on a function or procedure, and setting watchpoints to see when a variable or storage location gets changed.
- Two types of stepping operations: step-into and step-over. You can even step between functions or procedures written in different languages.
- Viewing the contents of global or local variables, call stacks, and data storage in specialized monitor panes.
- Quick restart of programs you debug repeatedly: Breakpoints and debugger window layouts for a given program are automatically saved between debugging sessions so that you don't have to set them again the next time.
- Attaching to an already running job or Java Virtual Machine on the iSeries system.

The Distributed Debugger supports any program written in the following languages:

- ILE RPG, COBOL, CL, C, C++, as well as Original Program Model RPG, COBOL, and CL.
- Java, when running the program under Windows, or when running on any iSeries system with OS/400 V4R2 or later. You can even debug servlets, JavaServer Pages^{TM}, and Enterprise JavaBeans on the iSeries system.

By using Object Level Trace along with the Distributed Debugger, you can trace and debug distributed applications running on various platforms including iSeries systems and Windows systems. Object Level Trace provides a graphical view of the different components of your distributed application, and you can click one or more elements to start debugging that component.

Two other debuggers are provided with the product:

1. You can use the workbench debugger for lightweight Java debugging on your workstation.
2. You must use the VisualAge RPG debugger to debug client-side VARPG applications on Windows 95 and 98. Debugger selection is handled automatically for you in VisualAge RPG based on your Windows platform.

Chapter 2. Prerequisites

Before you deploy Web applications using IBM WebSphere Development Studio Client for iSeries and any other relevant tools, you need to ensure that IBM WebSphere Application Server for iSeries and other services are properly configured on your iSeries system. This chapter describes the steps you need to take and provides links to appropriate documentation to prepare your iSeries system for operation.The minimum and maximum memory recommendations to run IBM WebSphere Development Studio Client for iSeries are 128 MB - 512 MB.

If you plan to undertake development work using the Remote Systems Explorer, Java development tools, or other ILE-focused tools, you need only know how to access your iSeries system to begin development, and you can skip the majority of this chapter. See Chapter 3, “Introduction to the Remote Systems Explorer” on page 19 to orient yourself with the workbench IDE.

Working with your HTTP server and WebSphere Application Server

Your Web-enabled iSeries applications use WebSphere Application Server to run the Java servlets and JavaServer Pages™ (JSPs) that communicate between the Web user’s browser and the iSeries programs or data. To serve your HTML pages and JSPs from the same iSeries system, you also need an HTTP server on that system. We recommend that you use the IBM HTTP Server powered by Apache. You can find documentation about how to use this server in the following place: IBM HTTP Server for iSeries Documentation Center.

WebSphere Application Server handles executing the JavaServer Pages, JavaBeans, and Java servlets that are generated for various processes. The primary documentation resources for IBM WebSphere Application Server for iSeries are available at the following Web sites:

WebSphere Application Server 3.5 Standard Edition documentation

WebSphere Application Server 3.5 Advanced Edition documentation

For 3.5 versions of WAS, as a minimum you need to carry out the steps under the section *Installing WebSphere Application Server*.

Becoming familiar with the IBM WebSphere® Application Server documentation, in particular the sections on *Installing WebSphere Application Server* and *Setting up multiple instances of the WebSphere administrative server*, is highly recommended.

IBM WebSphere Application Server Version 4.0 Advanced Edition for iSeries

IBM WebSphere Application Server Version 4.0 Advanced Single Server Edition for iSeries

For 4.0 versions of WAS, minimally you need to carry out the steps under the *Installation* link. Becoming familiar with the IBM WebSphere Application Server documentation, in particular the sections on *Installation* and *Setting up multiple instances of the WebSphere administrative server*, is highly recommended.

In order to prepare your system for Web application development and deployment, you need to complete the following tasks. iSeries-specific information is included, and you can follow the preceding links for information about the IBM HTTP Server and WAS servers:

- Become familiar with server port numbers (information provided below)
- Find the port for your HTTP instance
- Find the port for your WebSphere Application Server instance
- Start the HTTP server job
- Create your HTTP configuration
- Create your HTTP instance (optional; you can use the default instance)
- Start your HTTP instance
- Install the sample libraries (information provided below)
- Configure your WebSphere Application Server
- Start your WebSphere Application Server instance (information provided below)
- Change the HTTP port for your WebSphere Application Server instance
- Mapping a network drive to the iSeries system (information provided below)
- Start the WebSphere Administrative Console

Server port numbers

If you plan to use the default port numbers for WebSphere Application Server and the HTTP server; typically, the default ports are configured as follows:

- The default HTTP server instance uses port 80.
- The default WebSphere Application Server instance uses port 900.

If these are not the correct default port numbers, contact your system administrator to determine what these ports are. You can use these port numbers for your HTTP and WAS instances, or you can create your own. See the appropriate documentation for creating your own instances.

Installing the sample libraries

To work with the samples in this guide, you need to restore the WHOLESALE and WDSCLAB libraries on your iSeries system. You should do this even if you have already restored the libraries for a previous release of the product, because their contents are different. The instructions describe how the restore the WHOLESALE library. To restore the WDSCLAB library, repeat the instructions exactly except use the word ″WDSCLAB″ everywhere you see the word ″WHOLESALE″.

Note: The save files used to install the sample library are for use with a V4R5 or later iSeries system.

To restore the WHOLESALE library:

1. On the iSeries system, create a library to contain the save files. To create a new library in the emulator, use the CRTLIB command and follow the prompts.
2. Create a save file using the CRTSAVF command:

   ```
   CRTSAVF FILE(MYLIB/WHOLESALE)
   ```

   where *MYLIB* is the library you created.
3. On your workstation, open a Command Prompt window.

4. Change to the following directory: `cd x:\WDSC\Wdscsample` where *x:\WDSC\* is the IBM WebSphere Development Studio Client for iSeries installation directory.
5. On the command line, enter:`ftp` *`hostname`* where *hostname* is the name of your iSeries system.
6. Enter your iSeries user ID and password.
7. On the command line, enter `cd` *`MYLIB`* where *MYLIB* is the library you created to contain the save files.
8. Enter the following:

   ```
   bin
   put wholesale.sav wholesale
   quit
   ```
9. Back in the iSeries console, enter `RSTLIB` and press F4.
10. In the **Saved Library** field, enter `WHOLESALE`.
11. In the **Device** field, enter `*savf`.
12. In the **Library** field, enter *MYLIB*, where *MYLIB* is the library you created to contain the save files. This restores the WHOLESALE library on your iSeries system.
13. Enter `addlible WHOLESALE` to add the WHOLESALE library to the library list.

Note: See "Adding the sample library to your initial library list" on page 48 for information about changing your job description to add this library to your initial library list.

Starting your WebSphere Application Server instance

You need to restart your WebSphere Application Server instance each time it is stopped, and each time your iSeries system is restarted. Follow these steps:

1. Log onto the iSeries system and open a QShell prompt by typing the command `qsh`.
2. Change to the bin directory in the WebSphere Application Server product directory: `cd /QIBM/ProdData/WebAsAdv/bin`
3. Enter the following command:

   ```
   strwasinst -instance wasinst -http httpinst
   ```

   where:

   wasinst

   is the name of the WebSphere Application Server instance you are using, and

   httpinst

   is the name of the HTTP instance you are using.
4. When the command completes and you see a prompt character (usually a dollar sign), exit QShell by pressing F3.

Mapping a network drive to the iSeries system

Web development tools deploys your HTML, JSP, servlet, and other publishable files to the iSeries system on a shared network drive. If you can map a network drive to the /QIBM directory of your iSeries system using the NET USE command or the Map Network Drive command, you already have NET USE access set up correctly. Follow these steps to set up NET USE access to the iSeries system, depending on your operating system.

For Windows 98 and Windows Me:

1. Ensure that you have logged on to your Windows workstation using an ID that is valid on the iSeries system:
   a. Click the Start menu icon.
   b. If you see an entry **Log off** *username* on the Start menu, and *username* matches your iSeries user ID, skip to step 2.
   c. Click **Log off**.
   d. When prompted to log on to Windows again, enter your iSeries user ID and password.
2. Open a command prompt and type the command `NET USE`.
3. If you see a drive letter assigned to the /QIBM directory on your iSeries system, you are already set up for NET USE access.
4. Enter the following command: `NET USE * \\`*`MYHOST`*`\QIBM *`.
5. Enter your iSeries password when prompted.

For Windows NT and Windows 2000:

1. Open a command prompt and type the command `NET USE`.
2. If you see a drive letter assigned to the /QIBM directory on your iSeries system, you are already set up for NET USE access.
3. Enter the following command: `NET USE * \\`*`MYHOST`*`\QIBM /USER` *`USERNAME`* `*` where *USERNAME* is your iSeries user ID.
4. Enter your iSeries password when prompted.

If you can successfully connect to the network drive, NET USE access is set up properly. If instead an error message is displayed indicating problems connecting to the iSeries system, check with the system administrator that TCP/IP is properly started on the iSeries system. You can start it yourself if you have administrative privileges, by logging onto the iSeries system, typing `STRTCPSVR *NETSVR` and answering any prompts as needed. When this command completes, try the steps described above again. If you are unable to map a network drive to your iSeries system, contact the system administrator.

Chapter 3. Introduction to the Remote Systems Explorer

The Remote Systems Explorer (RSE) is the workbench perspective for iSeries development tools. Use this perspective to connect to your iSeries host, or to various other remote systems such as UNIX or Linux, and perform actions against objects.

You define your connections to an iSeries host through an RSE Communications connection, and manage those objects through a filtering system that presents information from your iSeries host in a tree view. A connection contains subsystems. Subsystems handle the task of retrieving remote information and executing commands. Filters are contained in subsystems and handle the tasks of organizing native files, Integrated File System (IFS) files, commands, and jobs. The RSE Communications layer also provides caching for rapid performance over a slow network, and you can also work offline.

Note: Subsystems and profiles in the Remote Systems Explorer are distinct from OS/400® subsystems and profiles.

Note: For more detailed information, see the specific component in the documentation for the Remote Systems Explorer.

Before you begin

You can complete the exercises only if following prerequisites are met. Prerequisites are discussed in more detail in Chapter 2, “Prerequisites” on page 15.

1. You have the Java Development Kit version 1.3 installed on the server. (It is installed on the client automatically.)
2. You have TCP/IP access to an iSeries host.
3. You have started the iSeries host servers with the command `STRHOSTSVR *ALL`.

This chapter is divided into following sections:

- “Managing connections and profiles within the Remote Systems Explorer” on page 20
- “Using the iSeries Objects subsystem” on page 22
- “Using the iSeries Commands subsystem” on page 25
- “Using the iSeries Jobs subsystem” on page 26
- “Using the iSeries Integrated File System subsystem” on page 26
- “Editing, compiling, running, and debugging a program” on page 27
- “Synchronizing the RemoteSystemsConnections project to a repository” on page 29
- “Troubleshooting” on page 30

Managing connections and profiles within the Remote Systems Explorer

This section describes how to connect to your iSeries host and how to create and manage multiple connections and profiles from within the explorer. All tools using the Remote Systems Explorer can share connection definitions for remote systems and their associated communications channel. By sharing the communications channel, you only need to sign on once; you have easy access to your job elements (such as your library list and environment variables); and you have shared cache for all tools in the server definition. The **Remote Systems** view, in the Remote Systems Explorer, appears as a typical tree view.

Configuring your first connection to an iSeries host

The first time you connect to your iSeries host, you need to define a profile. Profiles offer a way to group connections, share connections, or keep them private. You can also store and share profiles in a Concurrent Versions Systems (CVS) repository. Your first profile will be for your local workstation. As you complete the steps for your first connection, you can decide whether to use the personal profile you just created, or a team profile so that you can share resources and information with other people.

Placing all connection and filter data in a specific profile greatly aids in team support. If every member of a team makes their team profile active, then connections and filter pools can all be shared. Team synchronization makes all active team profiles available to team members. Because everything except your user ID is stored on a central server, any developer can obtain all profile resources from any workstation by retyping the user ID for use on a new workstation. If you choose not to make a profile active, it exists only on your personal workstation. To reduce collisions on synchronization, user IDs and the ordering of connections within a pool are stored locally on your workstation. See "Synchronizing the RemoteSystemsConnections project to a repository" on page 29 for more information.

To configure your first connection:

1. If the Remote Systems Explorer perspective is not already open, click **Perspectives > Open > Other > Remote Systems Explorer**. (The name of the perspective you are in displays in the title bar.)
2. In the **Remote Systems** view, click the plus sign beside **New Connection** to invoke the **New Connection** window.
3. Enter a name for your first profile and click **Next**.
4. Click the **Parent profile** drop-down list.
   - To share your resources, select the Team profile. In most cases you would probably use the team profile.
   - To keep your files private, select the profile you created in the previous step.
5. Enter a connection name. This name displays in your tree view and must be unique to the profile. You might find it useful to name your connection after your iSeries host, since you can use the Remote Systems Explorer to connect to many hosts.
6. Select an operating system from the **System Type** drop-down list. You would most likely select **iSeries**.
7. For **Host Name**, enter the name or TCP/IP address of your iSeries host, for example, PROD400.

8. Enter your user ID for the iSeries host.
9. Add a **Description**; the description appears in the **Properties** view after the connection is created.
10. Click **Finish** to define your system.

Note: To check your port number, see the **Properties** view of the Remote Systems Explorer. You will see that the port is ″0,″ which means that your RSE Communications server will pick any free port on the iSeries host. You can specify a specific port number if you need to. For example, to work with a firewall.

Connecting and disconnecting

After you create a connection to an iSeries host, you can easily connect and disconnect.

To connect, expand the new connection in the **Remote Systems** view to reveal your subsystems (click the plus control). To connect to any of them, highlight the subsystem, right-click, select **Connect**, and the IDW prompts you for your password. The IDE also prompts you for your password if you drill down through the various subsystems to view files.

To disconnect, right-click the subsystem in the **Remote Systems** view and select **Disconnect**, or close the workbench. The workbench notifies you of any outstanding communications requests. However, any tools launched from the workbench that run externally from the RSE Communications server, such as the debugger or CODE Editor, are not affected because they have their own communication connection.

Note: You can monitor and change the properties of your connection in the **Properties** view of the Remote Systems Explorer perspective. Some values are read-only, and you can change others, such as the description or the hostname. Although each Remote Systems Explorer subsystem maintains its own list of properties, three properties (connected or disconnected, port, and user ID) are shared among all iSeries subsystems. If you change any of these properties in one subsystem, the other subsystems reflect the change. Select a subsystem and check the **Properties** view to see the shared properties for all of your subsystems. For example, the ″Connected″ value is either ″Yes″ or ″No″ for all of your subsystems under one connection. If you have difficulties connecting and disconnecting from the host server, see “Troubleshooting” on page 30.

Creating a second connection

You can create another connection to access another remote system. You can also create a second connection to the same hostname with a separate job environment, for example, to use an alternate library list. If you want, you can use a different user ID for each connection. Simply click the plus sign beside **New Connection** in the **Remote Systems** view and follow the same steps as you did for your first connection, but this time select your already created profile (or the **Team** profile) and make sure to give your second connection a new name.

Creating a second profile

You might want to create a second profile for connections and filter data that belong to a particular version of a software release, and are shared by all team members. You might also want to create several personal profiles in order to partition your own material.

To create a second profile:

1. Click the Menu button ▼ on the toolbar for the **Remote Systems** view and select **New profile**.
2. Specify a name for the new profile and click **Finish**.

Now when you create a new connection, you can select your new profile to filter elements from the iSeries host.

Using the iSeries Objects subsystem

The Remote Systems Explorer provides Native File System (QSYS) support. The iSeries Native File System lets you query what objects are on your iSeries host and perform actions against those objects. Filters allow you to easily organize elements within your system. Use the *filter* function to list Native File System objects (such as libraries or files). When you access the list by expanding the filter, you can perform actions against remote objects.

The **Remote Systems** view exposes subsystems, filters, and items specified by each filter. This view lets you organize your filtered information in an easy-to-understand tree view. You can create filters for collections of libraries, objects, and source members. Each filter contains only one type of item.

Using filter pools

Before you begin creating simple filters, you need to understand the concept of filter *pools*. You can decide if you want to use simple filters or filter pools to organize the elements filtered from your iSeries host. If you have been using the workbench for some time, your workspace might contain too many filters to navigate easily. In this case, filters can be partitioned into pools, where one pool contains filters for many different elements, depending on your needs. For example, a filter pool could contain a combination of two library filters, one object filter, and six member filters, or, a filter pool could contain a series of object filters and nothing else. To see how filter pools work:

1. In the **Remote Systems** view, expand **iSeries Objects** to see a collection of filters for Libraries, Objects, and Members, plus a library list.
2. Right-click **iSeries objects** and select **New**. You have the option of creating a new **Library**, **Object**, or **Member** filter. You do not need to create anything yet; just observe what is available.
3. Click the minus sign beside **iSeries Objects** to close the list.
4. To illustrate the use of Filter Pools, click the Menu button ▼ on the toolbar for the **Remote Systems** view, and select **Show Filter Pools**.
5. Click the plus sign beside **iSeries Objects**, and you can now see your filters listed under **DefaultFilterPool**.
6. Right-click **iSeries Objects** and select **New**. Now, you see the option to create a new **Filter Pool** or **Filter Pool Reference**.

7. Right-click **DefaultFilterPool** and select **New**. The option to create a new **Library**, **Object**, or **Member** filter now resides within a specific filter pool.

Creating a library filter

Library filters retrieve and display a set of libraries from your iSeries host in the **Remote Systems** view. To create a library filter:

1. Expand **iSeries Objects**.
2. Click the plus sign in front of **Your libraries**. Or, if you are using filter pools, right-click your filter pool and select **New > Library filter**.
3. Give your filter a name and click **Add** to define the filter string.
4. Specify your Library filter string. Select a predefined filter string from the drop-down list, or, specify your own library or filter (for example, y* to retrieve all libraries starting with ″y″). You can also browse to locate libraries.
5. Click **Finish** in both windows.

Your newly created Library filter automatically expands, displaying the libraries.

Creating an object filter

Object filters retrieve and display a set of objects from your iSeries host in the **Remote Systems** view. To create an object filter:

1. Expand **iSeries Objects**.
2. Click the plus sign in front of **Your objects**. Or, if you are using filter pools, right-click your filter pool and select **New > Object filter**.
3. Give your filter a name and click **Add** to define the filter string.
4. Browse, specify or select from the **Library** drop-down list the library that contains your objects.
5. Specify your object filter string. The default is * (asterisk), and you can use up to 10 characters. This window defines your filter. If, for example, you enter ″t*″, the filter retrieves all objects with names starting with ″t″.
6. To define object types and attributes (optional), click **Add**. The **Add Object** window appears:
   - Specify or select both the object type and attribute from the lists to create your filter. You can define multiple combination of types and attributes as part of one object filter string. Thus, this list-box collects all the combinations that have been defined for this filter string. Click **OK**

     Note: You cannot use an asterisk for both the object type and the attribute.
7. Click **Finish** in both windows.

Your newly created Object filter automatically expands, displaying the objects.

Creating a member filter

Object filters retrieve and display a set of source members from your iSeries host in the **Remote Systems** view. To create a member filter:

1. Expand **iSeries Objects**.
2. Click the plus sign in front of **Your members**. Or, if you are using filter pools, right-click your filter pool and select **New > Member filter**.
3. Give your filter a name and click **Add** to define the filter string.
4. Browse, specify or select from the **Library** drop-down list the library that contains your source members.

5. Browse, specify or select from the **File** drop-down list the file within the library that contains the appropriate source members.
6. Specify your **Member** filter string. The default is * (asterisk), and you can use up to 10 characters. This window defines your filter. If, for example, you enter ″s*″, the filter retrieves all members with names starting with ″s″.
7. Use the check boxes to select whether you want to retrieve source members, data members, or both. You must select at least one.
8. To select a specific member type (optional), click **Add**:
   - Specify or select a member type from the drop-down list and click **OK**.
9. Click **Finish** in both windows.

Your newly created Member filter automatically expands to displaying the members.

Viewing properties of a library, object, or member

Drill down through your subsystems to expose libraries, objects, and members. In the Remote Systems Explorer perspective, a **Properties** view is open by default, showing information such as the name of the library/object/member, text, type, subtype, and so on.

Depending on the object type, you might also execute some simple actions, such as deleting, browsing, editing, renaming, and compiling. Use the Commands subsystem to perform additional actions against objects. You can directly edit the contents of property fields. If any errors are generated when you refresh, the **Commands** view displays error messages.

Launching the Table view

The **Table** view displays detailed information for each field, member, and object. To work with this comprehensive view of items, right-click any set of libraries or files under **iSeries Objects**:

- For libraries, select **Show in table > Basic** to show the library names, types, attributes, text, and status, or **Show in Table > Extra** to show the same attributes as Basic, plus last modified dates, creation dates, and sizes.
- For physical files, select **Show in table > Members** to display member names, types, attributes, text, and status, or **Show in Table > Fields** to show field names, records, types, lengths, and text.
- For display files, select **Show in table** to file names, records, types, lengths, and text.

Note: You can sort the entries by clicking on the column heading.

Changing and deleting filters

To change the filter string for a filter:

1. Right-click the filter and select **Change**.
2. On the filter window, highlight the filter you want to change, click **Change** to invoke the filter string dialog.
3. Edit the filter information and click **OK**.
4. From the **Change Filter** window, click **OK** again.

To delete a filter:

1. Right-click the filter and select **Delete**.

2. On the Delete Confirmation window, click **Delete**. Your workspace will automatically refresh to show that the filter has been removed.

Using the iSeries Commands subsystem

Use the Commands subsystem to:

- Define command sets
- Invoke programs and commands on an iSeries host
- View the output messages displayed in the **Commands** view

When you run a command, the **Commands** view displays all messages plus help for the messages. Messages also display under the command in the **Remote Systems** view. The iSeries Commands subsystem provides similar functions as the Command Shell in the CODE Editor.

Using predefined commands

For some of the frequently used commands, the subsystem provides you with a number of predefined command sets. You can use these command sets or create new ones of your own. To run the ADDLIBLE command set:

1. In the **Remote Systems** view, expand the **Commands** subsystem.
2. Right-click **Add library to library list** and select **Run**.
3. Enter the library name. For this example, enter ″WDSCLAB″ to place the restored library in your workspace. (Before this step, you should have already completed the prerequisite instructions for “Installing the sample libraries” on page 16.)

Creating your own command

If you want to add your own command set to the list:

1. From the Commands subsystem in the **Remote Systems** view, click the plus sign beside **Your command(s)**.
2. Enter a name for the command set, which describes what the command set does. This name appears in the **Remote Systems** view upon creation of the command set.
3. Click **Add**.
4. Enter the command you want to run. If you are not sure what the exact name is of the command you want, click **List** to display all of the commands available, and select from this list.
5. If applicable, select one of the check boxes for: **Normal, Batch** or **Interactive**. If you want to run the command in an interactive job, make sure to complete steps 1-3 of ″To run an interactive program″ on page 29 before you run the command. For more information on the different types of ways to run programs and commands, see “Running programs or commands” on page 28.
6. Click **Prompt** if you want to specify parameters to the command.
7. If you would like to have the command prompted when it is run, select the **Prompt when run** check box.
8. Click **Finish** in both windows.

Using the iSeries Jobs subsystem

At any time, a number of jobs are running on your iSeries host. You or other members of your team might be performing different actions at the same time. To keep track of jobs, the Remote Systems Explorer employs a Jobs subsystem that lets you identify subsets of server jobs and and view job information. The Jobs subsystem uses filters to select which jobs display in the **Remote Systems** view.

To create a filter:

1. From the **Remote Systems** view, right-click **iSeries Jobs** and select **New > filter**. Or, if you are using filter pools, right-click your filter pool and select **New > Filter**.
2. Give the filter a name and click **Add** to invoke the **Job Filter String** window.
3. Enter specifications for the job name, user ID, and job number. For example, ″NF*/jsandler/*″ retrieves all of jsandler's jobs that start with NF.
4. Check the **Active, Job Queue**, or **Output Queue** check boxes to filter jobs based on their status as well.
5. Click **Finish** in both windows.

When you click one of the jobs retrieved by the filter, the **Properties** view displays information such as the job name, number, user ID, and status.

In addition, the Remote Systems Explorer contains an **iSeries Job Log** view that displays the job log for active jobs. To view job logs and properties:

1. Right-click the job and select **Properties**.
2. Right-click the job and select **Display Job Log** to open the **iSeries Job Log** view.

If you right-click your job, you can perform the following actions:

- To end a job, right-click the job and select **End > Controlled** (which allows programs running in the job to perform their end-of-job cleanup) or **End > Immediate** (to be used if a job is frozen and the **Controlled** option fails).
- To hold a job, right-click and select **Hold**. If your job is in the *Scheduled* state, **Hold** ensures that the job does not start to run as it otherwise would according to job scheduling. If your job is in the *Active* state, **Hold** will pause the job. If your job is in the *Finished* state, **Hold** marks the spool files as *Held*, and certain operations on the output queue will bypass these *Held* files.
- To release a job from the *Held* state, right-click and select **Release**.

Using the iSeries Integrated File System subsystem

The Remote Systems Explorer implements Integrated File System support. The IFS provides a common interface to the different systems on your iSeries host. The file system contains two default filters: File systems and Root file systems. File systems contains all of the file systems you have access to. Root file systems represents one component of File systems, and lists all of the specific files and folders under the root of the IFS. You can treat the IFS in the same way that you treat iSeries files, except the location of the files is different.

Integrated File System filter information

The IFS filter lists files from specified areas on your host, and works in a similar way to Windows Explorer.

To create an IFS filter:

1. Right-click **IFS Files** and select **New > Filter**. Or, if you are using filter pools, right-click your filter pool and select **New > Filter**.
2. Give the filter a name and click **Add**.
3. Specify an IFS folder that contains the files that you want to filter (You can also browse to IFS folders).
4. You can filter your files by name and/or by type.
   - To filter by name, enter a specification in the **File name filter** field. For example, enter ″r*″ to retrieve all files with names starting with ″r″.
   - To filter by type, specify a type of file in the **File types filter** field or click **Select** to obtain a list of file types. You can select more than one type. For example, enter ″java,html″ in the field to find all Java and HTML files under a given folder.
5. Click **Finish** in both windows.

Your newly created IFS filter displays under the **IFS Files** subsystem.

Editing, compiling, running, and debugging a program

The Remote Systems Explorer is equipped to help you edit, compile, run, and debug your programs or files, all within one perspective. You no longer need to launch separate applications; instead, all functions are seamlessly integrated into one workspace.

Launching tools for editing

Use the Remote Systems Explorer to edit your source files directly with the integrated iSeries editor. The LPEX editor provides syntax highlighting and prompting for RPG, ILE RPG, and DDS source members.

You can edit source physical file members through the iSeries Objects subsystem using the LPEX editor or CODE editor. The LPEX editor lets you display, change, and save source members in your source physical files. CODE editor allows for more advanced capability, such as syntax checking and program verification, but these features will be added to the LPEX editor in future releases.

During development, your file is locked on the remote system so that others can not change it. Accordingly, if you attempt to edit a locked member, the workbench informs you that the file is in use, and allows you to open the member in browse (read-only) mode.

To edit your source members in the workbench:

1. Drill down through your subsystem in the **Remote Systems** view until you reach the source physical file member you want to work with.
2. Right-click and select **Open with > LPEX editor** to open the member in the integrated iSeries editor, or **Open with > CODE editor** to open the member in the classic CODE editor.
3. If you are using the LPEX editor, you can edit more easily with the source prompter. To open the source prompter, click the **Source Prompter** tab at the bottom of the screen, or click **Perspective > Show View > Other > iSeries > Source Prompter** and then click **OK**. Click anywhere inside your file and press F4 to activate the source prompter.
4. Make your changes in the editor.

5. You can either save your work and close the editor, or compile. See the next section for information on compiling.

Compiling a program

The right-click menu for source members includes a number of predefined names for compile commands. A compile name is an identifier for a compile command to be run on a remote system. For example, the compile label ″CRTBNDRPG″ tells the iSeries host to create a bound RPG program. Compile names are only associated with source members types: modules (*MODULE), programs (*PGM), and service program objects (*SRVPGM). For this reason, you will only see the options to compile when you right-click these types of elements.

There are two ways to compile: prompted and non-prompted. Both compile actions invoke submenu with a list of compile names. To compile source members from within the **Remote Systems** view, right-click a source member and select **Compile > <*name*>** or **Compile (No Prompt) > <*name*>** where <*name*> represents the command you want to run.

- When you use **Compile**, a window appears that allows you to view and change parameters for that command. Some of the parameters are already filled for you, based on the source member or object you right-clicked. After you alter the parameters, click **OK** to run the command.
- When you use **Compile (No Prompt)** , the command is performed on the server without prompting you first.

Note: Only the compile names associated with the specific member type are shown in the submenus. You can manage compile names and their source-type associations by right-clicking a source member and selecting **Compile > Work with Compile Commands** or **Compile (No Prompt) > Work With Compile Commands**. You can track the results of your compile from the **Commands** view.

Running programs or commands

You can run your programs and commands in three ways:

1. In the Remote Systems Explorer communication server job
2. In a submitted batch job
3. In an interactive job

Selecting the first option lets you run the program in the same job as the communications server. The advantage of this method is that you are notified when the program command ends. With basic batch and interactive jobs, you are not able to monitor the status as easily, however, you do not tie up your communication server. Batch jobs work as you would expect and do not require any initial setup. Interactive programs require a 5250 emulator, so you need to first run a CL command to associate an emulator with the Remote Systems Explorer communication server.

To run a program in the Remote Systems Explorer communication server job:

1. Right-click your program object and select **Run > Normal**. The **Commands** view provides feedback.

To run a batch program:

1. Right-click your program object and select **Run > Batch**. The **Commands** view console provides feedback.

To run an interactive program:

1. Click the Menu button ▾ on the toolbar for the **Remote Systems** view and select **iSeries > Start Daemon** to start the Communications daemon.

   Note: You only need to start the daemon once.
2. Open a 5250 emulator and sign-on to your remote system.
3. In the emulator, enter the command: `STRRSESVR <`*`connectionName`*`>` where *`connectionName`* is the name of your connection defined in the **Remote Systems** view. This associates the interactive job with the Remote Systems Explorer communications server.

   Note: The connection name is case sensitive and must already be defined in the Remote Systems Explorer.
4. Return to the workbench, and in the **Remote Systems** view, right-click your program object and select **Run > Interactive**. The **Commands** view provides feedback.

Debugging

You can easily debug your program from the Remote Systems Explorer by right-clicking a program object, job (under the Jobs subsystem), or .class file (under the IFS subsystem) and selecting **Debug**. This action invokes the Distributed Debugger and fills in the first page from the element you selected. See The IBM Distributed Debugger for AS/400 for more information.

Synchronizing the RemoteSystemsConnections project to a repository

The team support model centers on repositories that store version-managed resources on shared servers. You can synchronize your RemoteSystemsConnections project with various types of repositories, such as CVS (Concurrent Versions Systems) or Rational ClearCase, when you share profiles between team members. The original profile is only created once yet everyone can use it in his or her personal workspace.

The repository stream maintains a team's shared configuration of one or more related projects and their files/folders. The main developer and the team members can all synchronize their workspaces with the team stream, plus they can always download the latest configuration. The repository is designed to handle and resolve any possible conflicts with simultaneous team activity, and can merge changes in a rational fashion. This model supports groups of highly collaborative developers who work on a common base of files and frequently share their changes. It also supports developers who work offline for long periods, connecting to their repository only occasionally to synchronize with the stream.

Note: Before you synchronize your workspace with a repository, you must first have access to a CVS repository, be associated with a repository location, and have the following repository information: connection type, appropriate user name to use for synchronization, hostname, and repository path.

To download the latest work and synchronize your workspace with a CVS repository:

1. Click the Menu button ▾ on the toolbar for the **Remote Systems** view, and select **Team > Synchronize with Stream**.

2. Select a repository and click **OK**. This opens the Team perspective.
3. Click the **Repositories** tab (at the bottom of the **Synchronize** view).
4. Right-click in this view and select **New > CVS Repository Location**.
5. Use the information given to you by your CVS administrator to fill in the fields.
6. Click **Finish**.
7. Enter your password in the pop-up window and click **OK**.
8. Click **Perspectives > Open > Other > Remote Systems Explorer**.
9. Click the Menu button ▾ on the toolbar for the **Remote Systems** view, and select **Team > Properties**.
10. Right-click **RemoteSystemsConnections** and select **Properties**.
11. Click **Team** and click **Change**.
12. Click a repository name, click **OK**, and then click **OK** again on the main **Properties** window.
13. Click the Menu button ▾ on the toolbar for the **Remote Systems** view, and select **Team > Synchronize with Stream** to synchronize to the specified CVS repository. The **Synchronize** view becomes active.
14. Right-click your project and select **Release** to transfer the the profiles to the server, or **Catch up** to transfer the files to your local system.
15. Refresh the workbench to view the new profiles.

Troubleshooting

This section contains solutions to possible problems you might encounter when working within the Remote Systems Explorer.

Problem: I cannot connect to the host server.

If you are not able to connect to the host server through the Remote Systems Explorer:

1. Click **Window > Preferences**.
2. Expand **Remote Systems** and click **iSeries**.
3. Select the **Run RSE Communications Server on remote system** check box and try connecting again.

If you are still unable to connect to the host server, leave the above check box checked, and try connecting to the server manually:

1. Open an emulator, logon to your iSeries host and open a QShell prompt by entering the command `qsh`**.**
2. Change to the directory where the server files are by entering: `cd /QIBM/ProdData/DataStore/comm`.
3. Invoke a script by entering: `server.iSeries<`*`portnumber`*`>`

   Note: The port number is optional. If you decide to specify, make sure that it is a valid and unused port on the host machine, as well as the same port specified in the workbench for the subsystem.
4. (Optional) In the **Remote Systems** view of the Remote Systems Explorer perspective, right-click any of your subsystems and select **Properties**.
5. In the **Properties** view, find ″0″ for your port value.

6. Change this value to the port specified in your manual connection. This forces the client to bypass the automatic connection to the host server, and connect through your manual specification.

Problem: I cannot start my RSE Communications server.

The first thing you might want to try is starting the server manually:

1. Log onto the iSeries host and open a QShell prompt by entering the command `qsh.`
2. Change to the directory where the server files are by entering: `cd /QIBM/ProdData/DataStore/comm`.
3. Enter `server.iseries`.
4. The iSeries host will tell you which port it is using. Check to make sure that this port matches the port specified in the Remote Systems Explorer.
   - To find the port number, right-click one of your subsystems in the Remote Systems Explorer and select **Properties**. From the pop-up window, click **Info**. If the port does not match the one specified by the iSeries host, edit this field to include the proper port number.

If your RSE Communications connection still fails, the workbench will automatically determine what elements might be missing from your setup (e.g., required JDK level, correct hostname) plus write information to a log file. The workbench log file is located in: `\workspace\.metadata\.log` the Remote Systems Explorer log file is located in: `\workspace\.metadata\.plugins\com.ibm.etools.systems.core\.log`, and the host log file is located in `<`*`home_directory`*`>\.eclipse\RSE\rsecomm.log.` The home directory is set in the user profile, but typically this is `/home/<`*`userid`*`>`.

Note: These log files are only intended for development and support personnel.

Problem: My RSE Communications connection was successful, but my server still failed to connect.

This situation is quite unlikely, considering that a successful RSE Communications connection implies that the TCP/IP communication is fine and you have sufficient authority on the local workstation and server. If the problem is more specific, you can check the log file and determine if, for example, a port filtering firewall that exists between the client and the server has the JavaApplicationCall port open but not the specified RSE Communications port.

Chapter 4. Introduction to Java development tools

This chapter shows you how easy it is to work with Java files and create Java-based applications from a Windows workstation. Java development tools is equipped with specially designed wizards and views to help you import, export, compile, run, and debug your Java programs. The toolset is also equipped with a Program Call wizard that calls a program or service program procedure on an iSeries host, and can create Java beans and Program Call Markup Language (PCML) files, which the Web Services wizard can use to create a Web service.

Java development tools includes the IBM Toolbox for Java and three groups of presupplied beans: Data File Utility beans, Object List Beans, and Swing JFormatted beans. Use the Java Bean Examples wizard to port the presupplied beans into your workspace. These applications take advantage not only of iSeries data, but of the excellent performance characteristics of the iSeries Java Virtual Machine.

Before you begin

You can complete the exercises only if following prerequisites are met. Prerequisites are discussed in more detail in Chapter 2, “Prerequisites” on page 15.

1. You have TCP/IP access to an iSeries host.
2. You have started the iSeries host servers with the command `STRHOSTSVR *ALL`.
3. You have configured the desired server as a remote system.

This chapter is divided into the following sections:

- “Creating a Java project”
- “Using the Program Call wizard” on page 34
- “Porting Java beans into the workbench IDE” on page 36
- “Integrated File System support for Compile and Run” on page 37
- “Remote File System support for import and export” on page 37
- “Scenario: Exporting, compiling, running, and importing a Java application” on page 38

Creating a Java project

When you begin with Java development tools, you first need to create a project. Your Java projects contains Java code and an associated Java builder that compiles your source files when you change and save them. To create an empty Java project:

1. Open the Java perspective by clicking **Perspective > Open > Other.**
2. Click **Java** and then click **OK**.
3. Click **File > New > Project.**
4. Click **Java > Java Project** and then click **Next**.
5. Enter a name for your project and click **Finish** to create an empty Java project.

Using the Program Call wizard

The Program Call wizard helps you create the Java beans needed to invoke an iSeries program or procedure. The wizard prompts you for information regarding program or service program objects, plus the parameters of the program, and then generates the desired Java beans and a PCML file. This wizard generates two types of Java beans. One type can be used with Java development tools, and the other type, with Web Services.

Opening the Program Call wizard and loading your source

Before launching the wizard, you need to decide what types of Java beans you want to generate and how you want to use them. If you want to use them for a Web Service, you need create a Web project first. If you want to use them for Java development tools, you need to have a Java project. Use the project in which you want the wizard to generate the Java beans and PCML file.

To use the Program Call wizard:

1. In the **Java** or **Web** perspective, click your project to select it.
2. Click **File > New > Other** from the main menu.
3. Expand **iSeries** and click **Java > Program Call bean**.
4. Click **Next** to invokes the Program Call wizard.
5. In the **Java bean name** field, enter the name of the Java bean that you want the wizard to generate.
6. Click **Browse** to locate your Program object or Service program object. This automatically populates the **Program object**, **Library**, and **Program type** fields.
7. If your program type is *SRVPGM, you must input information in **Entry point** and select **void** or **integer** for **Return type**.
8. In the **Source Location** drop-down list:
   - **Browse iSeries native files** opens the browse window where you can select a program source from the iSeries native libraries.
   - **Browse iSeries IFS** opens the browse window where you can select a program source from the iSeries Integrated File System (IFS).
   - **Browse Workspace** opens the browse window where you can select a file from your current workspace.
9. Click **OK** to add this definition.

   Note: Click **View** to look at the source listing to help you determine what parameters are needed for the program or procedure.

Adding structures and parameters

At this point, you can add structures or parameters to your program or procedure. Before you add parameters, you must first specify a program (discussed in the previous section). If the program requires parameters, then you can define them with the **Add Parameter** function. A parameter can be of different data types: if it is a structure, you need to declare the structure first with the **Add Structure** function. A structure usually consists of variables, and you can use the **Add Parameter** function to add variables to a structure; hence, structures can be nested.

To add structures to your program:

1. While still in the Program Call wizard, click **Add Structure**.
2. Give the structure a name.

3. Define the fields accordingly (optional).
4. Click **OK**.

To add parameters to your program or add fields to your structure:

1. Click **Add Parameter**.
2. Give the parameter a name and define the fields accordingly. You can obtain information from a database field by clicking **Specify**. A window appears with a list of predefined connections. Expand a connection, or define a new connection.
3. You are prompted to log on to the server if you have not yet logged on.
4. Drill-down through the library list to locate a record or a field.
5. You can select a record or a field to add. If a record is selected, all the fields in that record are added as parameters.
6. Click **Add** and then click **Close**. This adds the fields to your PCML tree.
7. If you want to define this parameter *as* a structure:
   a. Select **structure** from the **Data Type** drop-down list.
   b. Select an already defined structure from the **Structure name** drop-down list.
8. Any parameters created from a database can synchronize with that database. In essence, if you suspect that fields have been changed on the host, click **Synchronize** or **Synchronize All** to update the parameter locally.

Importing elements from a PCML file

You might want to import a structure or a program from an external PCML file, an existing PCML file created earlier with this wizard, or one that was generated by the ILE RPG or ILE COBOL compiler, or one that you created. After you import, you can add various components to your new PCML file.

To import elements:

1. Click **Import PCML**. This invokes an **Import PCML** window.
2. From the **Select PCML file from** drop-down list:
   - Select **Workspace** to import a PCML file from your IDE workspace.
   - Select **Remote File System** to import a PCML file from a remote server, which can be from the iSeries IFS.
3. After selecting the PCML file, the tree-view of the file displays so that you can select programs or structures from the list. If the selected program references a structure, then that structure will also be imported even if it is not selected.
4. Click **OK**.
5. In the main wizard page, click **Next** if you have finished adding programs, parameters, and structures.

Creating Java beans and the associated PCML file

The second page of the wizard helps you specify where and what types of Java beans you want the wizard to generate. The wizard can generate two types of beans. One type is a regular Java bean, which any Java application can use. The other type is an specialized Java bean that the Web Services wizard can use to create a Web service.

Two classes are generated by the Web Services type, the first being the Java bean class prefixed with the word ″Services″, and the second being the class *required* by the Java bean class, which is suffixed by the word ″Result″.

The wizard generates one Java bean per program, yet all of them reference the same PCML file. The default PCML file name is the program call Java bean name of the first program you defined.

Note: The beans also reference classes in jt400.jar and wdt400rt.jar. The classpath of the project where the PCML file and Java beans are generated is updated automatically to point to these jar files, which are located in the runtime directory of the ″com.ibm.etools.iseries.toolbox″ and ″com.ibm.etools.iseries.webtools″ plug-ins. If you want to deploy these files outside of the workbench, or move them to another project, make sure to set the classpath to point to these jar files.

To generate the PCML and Java bean files in the second page of the Program Call wizard:

1. Use the **Folder** and **Package** fields to specify the desired location for the PCML file and Java beans. Pop-ups are available if you click **Browse**.

   Note: Before clicking **Finish**, the wizard lists the files that will be created.
2. Click **Finish**.

Launching a prefilled Program Call wizard

When you have a PCML file in your workspace, you can choose to launch the wizard again with data prefilled from a specific PCML file. To launch a prefilled wizard:

1. From your workspace, right-click the PCML file and select **New > Other**.
2. In the window, click **iSeries > Java > Program Call bean**. This launches the wizard with prefilled data from the selected PCML file. See the previous section for instructions on creating the PCML file.

Porting Java beans into the workbench IDE

Java development tools includes the IBM Toolbox for Java and a number of presupplied beans for your use. Use the Java Bean Examples wizard to view examples of these Java beans. If you want to access these Java beans or classes in the IBM Toolbox for Java, you need to include the iseriesut.jar and jt400.jar in the classpath.

Use the Java Bean Examples wizard to:

- preview examples from the wizard without loading any classes into the workbench IDE
- create a new project in the workspace and import the example source

To run examples from the Java bean wizard:

1. Click **File > New > Other**.
2. From the window, expand **Examples** and select **iSeries > Java Bean Examples**.
3. Click **Next**.

   Note: The Java beans Examples wizard lists many Java beans. Click various beans to read a description. You can preview as many examples as you want and the wizard does not add anything to your workspace.
4. Select the **FormManager** Bean and click **Run** to preview the example in the wizard.
5. After running the example, close it and click **Next**.

6. Specify a name for the project and click **Finish**. The wizard creates a Java project and imports the example source into this project. You can then learn how to use these Java beans by studying the source code of these examples.

Integrated File System support for Compile and Run

The workbench IDE provides you with a specialized iSeries view to help you compile, run, and debug Java programs.

Compiling your Java program

To compile your Java program:

1. Click **Perspectives > Show View > Other**.
2. Expand the **iSeries Java** file, select **iSeries Compile and Run View**, and click **OK**.
3. To populate the **Working folder** field, click **Browse** to select a directory in the pop-up window.
4. Click **Advanced Options** to fine tune your run/compile (optional) and click **OK**.
5. Click the **Create iSeries Java Program** tab.
6. Use the left and right panes to select the class(es) to compile.
7. Click **CRTJVAPGM** to create the programs. The log provides confirmation of success or informs you of any errors.

Running your Java program

If your compile is successful and you want to run your Java program:

1. In the **iSeries Compile and Run View** view (that you should already have open), click the **Run** tab.
2. In the **Class** field, enter the name of the class file you just compiled, or click **Browse** to locate the file in the IFS directories window.
3. Click **Run** to start the remote invocation. The **Console out** field displays output, program statements, or any errors.
4. Use the **Console in** field to enter information for the program (if the program asks for input).

Remote File System support for import and export

Java development tools provides you with specialized wizards that let you import Java files to, and export Java files from the workbench. Hence, these elements can be added to, or taken away from your workspace.

To import files to the workbench:

1. Click **File > Import** from the menu.
2. Click **Remote File System** and click **Next**.
3. Click **Browse** beside the upper **Folder** field to select a directory from a configured server in the **Browse for Folder** window. You might be prompted to sign on.
4. Use the left and right panes to select the resource(s) to import.
5. In the lower **Folder** field, specify the destination for the imported resources or click **Browse** to select from the **Folder Selection** window.
6. Select either of the **Options** check boxes if they apply.

7. Click **Finish** to complete the transfer.

To export files from the workbench:

1. Click **File > Export** from the menu.
2. Click **Remote File System** and then click **Next**.
3. Use the left and right panes to select the resource(s) to export.
4. In the **Folder** field, specify the destination for the exported resources or click **Browse** to select from the Remote File System.
5. Select either of the **Options** check boxes if they apply.
6. Click **Finish** to complete the transfer.

Scenario: Exporting, compiling, running, and importing a Java application

This scenario shows you how to work with a simple Java-based application in the workbench IDE. In this scenario, you will learn how to:

- "Create a small Java project"
- "Run the application locally" on page 39
- "Export the application and delete from workspace" on page 39
- "Run the application remotely" on page 40
- "Import the Java project back to your workspace and validate" on page 40

Before you begin

You can only complete this scenario only if the following prerequisites are met.

1. You have TCP/IP access to an iSeries server.
2. You have started the iSeries host servers with the command `STRHOSTSVR *ALL`.
3. You have configured the desired server as a remote system.

Note: You will work with the Integrated File System for this scenario. For that reason, you need to have a directory in this file system where you can try things out; i.e., export files to this directory. If you do not have your own directory in the Integrated File System, you can use a temporary directory, or create a directory from the Remote Systems Explorer.

Create a small Java project

In this exercise, you create an empty Java project and save one small Java file inside the project:

1. Click **Perspective > Open > Other > Java**.
2. Click **File > New > Project.**
3. From the **New Project** window, select **Java > Java Project** and click **Next**.
4. Name your project ″example″.
5. Click **Finish** to create an empty Java project.
6. When the project appears in the **Packages** view of your workspace, right-click the project and select **New > Package**.
7. Enter ″mypackage″ in the **Package** field, and click **Finish**.
8. Right-click ″mypackage″ and select **New > Class**.
9. In the **Name** field, enter ″World″.

10. Click **Finish**. This creates an empty Java file called ″World″ which automatically displays in the editor. The file should contain:

    ```
    package mypackage;
    public class World {
    }
    ```

11. In between the two brackets, paste in the following lines:

    ```
    public static void main(String[] args) {
       System.out.println("Hello World!"); //Display the string.
    return;
    }
    ```

    Now, the entire file should look like this:

    ```
    package mypackage;
    public class World {
    public static void main(String[] args) {
       System.out.println("Hello World!"); //Display the string.
    return;
    }
    }
    ```

12. Click **File > Save World.java**.

Run the application locally

Before you export the application, verify its generation locally:

1. While still in the **Java** perspective, click the ″World″ file to select it.
2. Click the arrow beside ″run″ icon in the taskbar and select **Run > Java Application**. This invokes the Debug perspective and displays ″Hello World!″ in the console at the bottom of the screen, letting you know that your application works locally.
3. Close the Debug perspective.

Export the application and delete from workspace

Java development tools for iSeries provides you with specialized wizards that let you export files from the workbench to an iSeries machine.

To export files from the workbench:

1. Return to the Java perspective.
2. In the **Packages** view, click ″example″ to select your newly created Java project.
3. Click **File > Export** from the menu bar.
4. Click **Remote File System** and then click **Next**.
5. In the left pane, select the check-box beside **example**.
6. In the **Folder** field, click **Browse** and navigate to the following directory in the IFS:

    ```
    <iSeries server>/<home directory>
    ```

    where *iSeries server* is the name of your iSeries server, for example, PROD400. Select your home (or temporary) directory and click **OK**.
7. Click **Finish** to complete the transfer.
8. Back in the **Packages** view of the **Java** perspective, right-click the ″example″ project, select **Delete** to delete from your workspace, and click **Yes** in the confirmation window.

Note: Deleting the project from your workspace is only for the purposes of this scenario - to see how you can import an application into a blank workspace. In everyday development, you do not need to delete your project when exporting or importing.

Run the application remotely

Now that you have exported the application to the IFS and deleted the original application from your workspace, you can run the application on the iSeries server right from your workstation.

1. Click **Perspectives > Show View > Other**.
2. Expand the **iSeries Java** file and select **iSeries Compile and Run View** You might have to expand this view to see all of the buttons.
3. Beside the **Working** field, click **Browse** to locate and select the directory where your project resides:

   `<iSeries server>/<home directory/<example>`
4. Click **example** and click **OK**.
5. Beside the **Class** field, click **Browse** and navigate to:

   `<iSeries server>/<home directory/example/mypackage/World.class>`
6. Click **World.class** and click **OK**.
7. Click **Run** to start the remote invocation. The **Console** displays "Hello World!", letting you know that your application has been exported successfully and can function remotely.

Create an empty Java project

Before you import the application back into your workspace, you need to create an empty Java project to accept the imported code:

1. Click **Perspective > Open > Other.**
2. Click **Java** and then click **OK**.
3. Click **File > New > Project.**
4. Click **Java > Java Project** and then click **Next**.
5. Name your project "container".
6. Click **Finish** to create an empty Java project.

Import the Java project back to your workspace and validate

Importing the project shows you the import capabilities of the Workbench IDE.

1. Click **File > Import** from the menu.
2. Click **Remote File System** and click **Next**.
3. Beside the upper **Folder** field, click **Browse** and navigate to your home directory in the IFS:

   `<iSeries server>/<home directory/<example>`

   where *iSeries server* is the name of your iSeries server, for example, PROD400.
4. Select **example** and click **OK**.
5. Select the check box beside the "example" folder in the left pane of the window. If you expand the folder and click on any of the files, you can see that they are all automatically selected in the right pane.
6. Beside the lower **Folder** field, enter "container".
7. Click **Finish** to import all the source code into your empty project. Click **Yes** if you are prompted to overwrite the classpath.

8. In the **Packages** view, expand **container > mypackage**, and click **World.java** to select it.

9. Click the arrow beside ″run″ icon in the taskbar and select **Run > Java Application**. This invokes the Debug perspective and displays ″Hello World!″ in the console at the bottom of the screen, letting you know that the imported application functions the same as before.

Chapter 5. Introduction to Web development tools

This chapter shows you how, with little or no knowledge of Java, HTML, JavaServer Pages^{TM}, or Program Call Markup Language (PCML), you can create Web applications on your iSeries system that interact with your iSeries databases through existing ILE programs. These Web applications can take advantage of your legacy iSeries data, of the excellent performance characteristics of the iSeries Java Virtual Machine, and of your existing skill in ILE development. Such applications can be generated very quickly with a minimum of programmer interaction, provided the ILE business logic is already in place on the iSeries system.

Web development tools give you the ability to create and maintain resources for Web applications in Web projects, which typically contain servlets, JavaServer Pages, Java files, HTML pages or images, and any associated metadata. Web Projects are always imbedded in Enterprise Application projects, since the Web development tools wizards that let you create a Web project or import a WAR file require that an Enterprise Application project be specified. When you complete tasks with the wizards, they will update the application.xml deployment descriptor of the specified Enterprise Application project so that your Web project is defined as a module element.

The iSeries extensions to Web development tools include a Web Interaction wizard that links the Design-Time Control (DTC) fields for a Web page to parameters of your Program Call and defines the Program Call parameters without the need to deal directly with JavaServer Pages, JavaScript code, or servlet code. The extensions also include all existing DTCs available in the classic WebSphere Studio for iSeries, and a new table Design-Time Control that helps you construct tables with multiple columns consisting of input fields that can be mapped onto host program input and output parameters in the Web Interaction wizard.

The main project folder for your Web application contains all related development objects necessary to create the application. This folder structure maps to the Web Application Archive (WAR) structure and contains the following elements:

- **source** - the project's Java source code for classes, beans, and servlets
- **webApplication** - all Web resources, including HTML, Java files, JavaServer Pages^{TM}, servlets, graphics, and any other related files
- **theme** - style sheets and objects
- **META-INF** - the MANIFEST.MF file, used to map class paths for dependent JAR files that exist in other projects within the same Enterprise Application project
- **WEB-INF** - the supporting Web resources for a Web application, including the web.xml file and the classes/lib directories
  - **web.xml** - the standard Web application deployment descriptor
  - **/classes** - servlets, utility classes, and Java compiler output
  - **/lib** - supporting .jar files referenced by your Web application

Before you begin

You can complete the exercises only if following prerequisites are met. Prerequisites are discussed in more detail in Chapter 2, “Prerequisites” on page 15.

1. You have TCP/IP access to an iSeries host.
2. You have started the iSeries host servers with the command `STRHOSTSVR *ALL`.
3. You have configured the desired server as a remote system.

This chapter is divided into the following sections:

- “Creating a Web project” on page 44
- “Creating a graphical user interface” on page 44

Creating a Web project

The J2EE model defines a Web application directory structure that specifies the location of Web content files, class files, classpaths, deployment descriptors, and supporting metadata. The Web project hierarchy mirrors that of the Web Archive (EAR) structure, defined in the J2EE model. To create a Web project:

1. Click **Perspective > Open > Other > Web**.
2. Click **File > New > Web Project**. This initializes the **Create a Web Project** wizard.
3. Enter a name for your project.
4. Uncheck **Use default location** and click **Browse** if you would like to specify where the project is kept.
5. Edit the Enterprise Application project name dialog if you would like to specify your own name for the EAR file.
6. The context root field is automatically populated when you enter a name for the project, which also populates the default location for the project. The context root is the the top-level directory of the Web application, for deployment to a Web server.
7. Click **Next** if you want to configure module dependency information. For example, you can include servlets, JSPs, Java files, static documents, and any associated metadata you want to incorporate into the user interface for your Web project.
8. Click **Finish** to create an Web project containing the proper folder structure, along with a web.xml file that describes the current project.

Creating a graphical user interface

You can take two approaches when creating Web input and output pages. You can either create your own, or let the Web Interaction wizard generate the pages for you. In either case, you can insert specific iSeries Design-Time Controls on your pages to enhance the input and output functionality of your Web interface. The Web Interaction wizard adds the javascript code and creates the servlets that the user interface Web pages need to communicate with your business processes.

When you take the second approach and use the Web Interaction wizard to create the JSP files and the servlets, the output contains a minimum amount of function and usability. Once the files are created, you can edit them to modify page display properties, add images, and update the text on the page and the labels on the buttons. Then you can rerun the Web Interaction wizard to regenerate the javascript and servlets to account for your changes. Although the second approach is simpler, the advantage of the first approach is that you can make your JSP file more advanced and sophisticated before creating your servlet.

When you take the first approach and create your own JSP file as a user-input page, you need to insert a form on the page to contain all input fields and Web application controls. This ensures that the Web Interaction wizard recognizes these values as input parameters for communication with your business processes.

For detailed information on how to use the wizards, see the specific iSeries Web development tools component in the help navigation.

Chapter 6. Introduction to the WebFacing Tool

You are now able to work with the WebFacing Tool in the workbench to convert your DDS display files to JSP files and JavaBeans, suitable for Web front-ends. Use the WebFacing wizard to select DDS source members and a Web style for your application. After you convert your application and invoke the application from your browser, the iSeries WebFacing server intercepts all green-screen display file calls and directs them to your browser. The WebFacing Tool is ideal for programs you want to deploy quickly to the Internet or your company intranet, or programs for which you plan to maintain Web and green-screen interfaces to the same business logic.

After you have completed the steps of setting up your iSeries host and your workstation for WebFacing, you can proceed with converting your DDS source. The easiest way to convert DDS code is to use the workbench WebFacing perspective.

First, you use the WebFacing Tool to generate the record beans and JSPs required for the Web interface of the application, from the DDS source. Next, you can then test your application with the integrated server and Test environment. When you are satisfied with the results, you then create the application resource in the WAS Console. Finally, your application is accessible from any Web browser on the network. Three servers are involved in enabling the Web-based user interface at run time:

- The WebFacing run-time server intercepts read and write operations from your iSeries application
- WebSphere Application Server runs the JSPs and JavaBeans that provide the Web interface
- IBM HTTP Server handles delivering content to the user’s browser and returning user input to WAS and the WebFacing run-time server (You only need to be concerned with this server if you have a separate instance running on the iSeries host.)

Before you begin

You can only complete the exercises if following prerequisites are met. Prerequisites are discussed in more detail in Prerequisites.

To convert an application and work with the integrated server and Test environment, ensure that:

1. The WDSCLAB is restored to your iSeries system.
2. The WDSCLAB is in your iSeries library list. See Adding the sample library to your initial library list for more information.

To be deploy and test an application from a stand-alone IE 5.0 browser, in addition to the previous instructions for the Test environment, you need to ensure that:

1. All the products listed in the introduction are installed.
2. You have created or have access to an HTTP server configuration and instance on the iSeries system, and the instance is running.
3. You have created or have access to a WebSphere Application Server instance on the iSeries system, and the instance is running.

4. You have NET USE access to the iSeries system, and have mapped the `/QIBM` directory of the iSeries system to a drive letter on your Windows system.
5. You know the port numbers for the HTTP and WebSphere Application Server instances on the iSeries system.

Note: Microsoft Internet Explorer 5.0 or higher is a requirement.

This chapter is divided into the following sections:

- "Adding the sample library to your initial library list" on page 48
- "Loading the DDS source" on page 48
- "Generating the JSPs and JavaBeans" on page 49
- "Testing your application" on page 50
- "Exporting your application" on page 50
- "Launching your application from a browser" on page 51

Adding the sample library to your initial library list

Before you load the DDS source, make sure that the WDSCLAB is in your initial library list. When using the WebFacing Tool, the initial program defined in your user profile is not run. Therefore, you need to change the initial library list setting of your job description (associated with your user profile) to control the library list used by your application. To include the library:

1. Open a 5250 emulator and sign-on to your iSeries host.
2. Enter `dspusrprf <`*`userid`*`>` to display your user profile.
3. Scroll through your profile until you find the lines for `Job description` and `Library`. (`Library` appears right after `Job description`.)
4. Make sure that the job description is yours or one used by your team. If not, create your own job description and associate it with your user profile. If you need assistance, see your system administrator.
5. Return to the main menu and enter `chgjobd` to change your job description.
6. In the **Job description** field, enter your user ID.
7. Press F10 to access additional parameters.
8. Scroll through your job description until you find the line `Initial library list`.
9. In the field underneath the line that says `+ for more values`, enter `WDSCLAB` to save your new job description.

Loading the DDS source

When you load the DDS source for an application into the WebFacing Tool, you can choose to generate the WebFacing code for it immediately. For a small application such as this soccer program, you would normally do this by clicking the **...proceed with conversion** radio button on the final page of the New WebFacing Project wizard. In this scenario you will perform the conversion in a separate step, to reflect how conversion might proceed with a more complex application.

When you create a project without converting it immediately, the DDS source members are not converted, but information about them is stored in the project. After completing the steps in this section, you can switch to the **WebFacing Projects** tab and expand the soccer folder and the DDS folder below that, to see the record names for the DDS source you imported.

1. From the workbench, click **File > New > Project.**
2. Click **WebFacing > WebFacing Project** and then click **Next**.
3. In the **Specify a new project for WebFacing** page, give your project a unique name, such as `soccer`.
4. If you want to supply a different destination directory for your project than the default, uncheck the **Use default location** box and click **Browse** to select a directory. Then click **Next**.
5. In the **Connection** selection box, select the connection that applies to the iSeries host that contains the WDSCLAB library.
6. In the **Library filter** field, enter `WDSCLAB`.
7. In the **Member types** drop-down list, select **DSPF MNUDDS**.
8. Click **Refresh list** (you might be prompted to sign-in).
9. Expand the library until you see the `GROUPIINQY` DDS source member. Click it and press the right arrow button to add the source member to the list of sources to convert.
10. Click **Next**.
11. The **Select UML source members to convert** page works exactly like the **Select display file source members to convert** page. In future, use this page of the wizard if you have UML help files you want to convert.
12. Click **Next**.
13. On the **Specify CL Commands** page, enter a label in the **Command label** field that signifies the command you want run on the iSeries host. For example, `Try out the soccer program`. This label is the text for the hyperlink that invokes the converted application from the browser.
14. In the **CL command** field, enter the actual command. For example, `CALL WDSCLAB/GROUPIINQY`. This is the command the WebFacing run-time server uses to start your application. Click **Add** and click **Next**.
15. Select a web style for the look and feel for your pages and click **Next**.
16. Select **No, I only want to create the project now** and click **Finish**.

Generating the JSPs and JavaBeans

For a complex application involving many DDS screens, you might want to convert DDS source members in small batches, so that you can focus on making the conversion of each page go smoothly. The WebFacing Tool does not fully support all DSPF keywords, so an incremental approach in larger conversions helps you avoid having to deal with long lists of conversion messages about what DDS specifications might need to be changed. In this section you convert the DDS source for the soccer application.

1. In the **WebFacing** perspective of the WebFacing Tool, expand the project tree until you find the `GROUPIINQY` file. Right-click the file and select **Convert**.
2. Log into the iSeries system if prompted, then wait for conversion to complete.
3. If you want to customize the look of your application in future, switch to the Web perspective to edit the JSPs and CSS files. The PageBuilder.jsp under the styles directory contains the main document that the converted application is included into. This is a good place to begin to customize. You can find the JSPs for each record in the RecordJsps directory. (See the documentation for Web development tools for more information about customization.)

Testing your application

Before you export and deploy your Web application, take advantage of the WebSphere Application Server run-time environment to test one or more Web projects. This tool simulates the finished product in an integrated browser so that you can easily alter your files and track the results without having to re-deploy each time.

To test your Web application:

1. In the navigator view, expand your project and right-click your index.html file.
2. Select **Run on server**. The Server development tools feature automatically completes the following tasks for you:
   - launches the Server perspective
   - creates a server project with `Servers` as the default name
   - creates the server instance with `WebSphere v4.0 Test Environment` as the default name
   - creates the server configuration with `WebSphere Administrative Domain` as the default name
   - sets the server instance to use the server configuration
   - adds your project to the server configuration
   - starts the server instance
   - opens the **Servers**, **Debug**, **Processes**, **Console**, and **Variables** view
   - displays the file in the integrated Web browser

Note: When running the Test environment, the server is running against the resources that are in the workbench. Therefore, you can reconvert your DDS, or add, change, or remove a resource from the Web project and the server does not have to be restarted in order to picks up these changes.

Exporting your application

After you are satisfied with the results in the Test environment, you can export your WebFacing project to the WebSphere Application Server, start your application, and then launch from a browser. Depending on the version of WebSphere Application Server that you are using, you need to do these activities in slightly different ways. Provided are links to proper documentation, an outline of the different processes, and a simple description of how to export (that you can modify depending on the version of WAS that you are using).

For information about starting Web applications for WebSphere Application Server versions 3.5 and 4.0, see the documentation sites for these releases.

WebSphere Application Server 3.5 Standard Edition for iSeries

WebSphere Application Server 3.5 Advanced Edition for iSeries

IBM WebSphere Application Server Version 4.0 Advanced Edition for iSeries

IBM WebSphere Application Server Version 4.0 Advanced Single Server Edition for iSeries

For WebSphere Application Server version 3.5 Standard and Advanced Editions:

1. Make sure you have NET USE access to the iSeries system, and have mapped to a drive letter on your workstation. See "Mapping a network drive to the iSeries system" on page 17 for more information.
2. Using the export wizard, export the files and directories of your WebFacing project to the iSeries shared drive. If you have mapped the /QIBM iSeries Integrated File System (IFS) directory to the drive letter X, then the directory you would export your project to would be: X:\UserData\WebASAdv\<*instance name*>\hosts\default_host.
3. Start the WebSphere Application Server version 3.5 administrative console (using the documentation provided in the above link).
4. Create a corresponding Web application with the WebSphere Application Server administrative console and start the application in the console.

For WebSphere Application Server version 4.0 Advanced Edition:

1. Make sure you have NET USE access to the iSeries system, and have mapped to a drive letter on your workstation. See "Mapping a network drive to the iSeries system" on page 17 for more information.
2. Using the export wizard, export the .WAR file to a temporary directory on the iSeries IFS.
3. Start the WebSphere Application Server version 4.0 administrative console (using the documentation provided in the above link).
4. Create a new Web application with this console, and make sure to import the .WAR file that you exported to the local file system.
5. Start the Web application using the administrative console.

For WebSphere Application Server version 4.0 Advanced Single Server Edition:

1. Using the export wizard, export the files and directories of your WebFacing project as a .WAR file to your personal folder on the IFS.

   Note: To save time, you can opt to export the files to your local file system, and then copy them over to the IFS:
2. Access the browser-based administrative console.
3. Create a new Web application with this console, and make sure to import the .WAR file that you exported to the local file system.
4. Start the Web application using this console.

Launching your application from a browser

After you start your application, it can be accessed through a browser with an address of the form: `http://host_name:<hostport>/<application_context>/` For example, if your server is PROD400, your port is 12055 and the Context root value that you entered while creating the Web application in the administrative console is `webfacingproj`, then you would access the application with an address like: `http://PROD400:12055/webfacingproj`.

Chapter 7. Debugging WebSphere Application Server Applications

You can use the Distributed Debugger to debug servlets, Java Server Pages, and Enterprise Java Beans that are deployed through WebSphere Application Server. Once you have loaded your program in the debugger, the debugger features you can make use of are as follows: stepping and running, setting breakpoints, examining variables and the call stack. The most challenging aspect of debugging iSeries Web applications is learning how to start the debug session. This chapter describes how to debug a servlet, which is part of your WAS application, using Object Level Trace (OLT). You can also debug applications by attaching directly to the running job for a WAS application server. See the debugger documentation for more information.

Object Level Trace (OLT) is typically the easiest way of starting to debug WAS applications. You set debugging options in your WAS application server, then start your application in a Web browser; each time a servlet, EJB, or JSP method is called that is in the debug classpath, OLT notifies you and gives you the opportunity to start debugging that method. If you choose to debug, the debugger is launched and execution stops at the first debuggable statement in the method you selected. If you choose not to debug, the application continues until the next servlet, EJB, or JSP method is called.

Introduction

The scenario in this chapter uses Object Level Trace or attaching directly to an iSeries job number to start debugging sessions for Web applications:

- In “Debug the snoop servlet” on page 55 you use OLT to start debugging the snoop servlet, a simple Web application that does not call any other debuggable classes.

The scenario involves using the following products and components:

- IBM WebSphere Development Studio Client for iSeries — Object Level Trace and Distributed Debugger
- IBM WebSphere Administrative Console
- IBM WebSphere Application Server for iSeries and IBM HTTP Server for iSeries

Object Level Trace is *not* an appropriate tool for debugging Web development tools applications that call iSeries ILE programs. OLT only helps you debug the generated servlets or JSPs for these applications, and because this is generated code there is no point in debugging it. Usually you want to debug the host ILE logic in such cases, and you can do this starting in the debugging JSP that is generated for your Web application.

Before you begin

You can only complete the scenarios in this chapter if the following prerequisites are met. Prerequisites are discussed in more detail in Chapter 2, “Prerequisites” on page 15.

1. You have TCP/IP access to an iSeries server.
2. You have started the iSeries host servers with the command `STRHOSTSVR *ALL`.
3. You have configured the desired server as a remote system.

4. You have created or have access to an HTTP server configuration and instance on the iSeries system, and the instance is running.
5. You have created or have access to a WebSphere Application Server instance on the iSeries system, and the instance is running.
6. You know the port numbers for the HTTP and WebSphere Application Server instances on the iSeries system.

The remaining sections of this chapter describe how to perform the following tasks:

1. Set up WAS Console to enable tracing and debugging of standard WAS applications or applications created using WDSC.
2. Start the Object Level Trace viewer and set options in it to enable tracing.
3. Select and start to debug objects.

Set debugging options in WebSphere Administrative Console

Before you can start tracing and debugging method calls originating from your iSeries system, you need to enable debugging in the appropriate application servers in your WebSphere Administrative Console. For applications such as the snoop servlet, the application server is the default server. You also need to specify a classpath to determine what directories WAS looks in for applications to debug. Classpaths are indicated below. Note that you can specify several directories separated by colons (:) if you want to debug more than one Web application; do not use semicolons (;) as this may prevent you from debugging or even running your application.

- For the default server (to enable debugging of the `snoop` servlet), specify `/QIBM/ProdData/WebAsAdv/hosts/default_host/default_app/servlets`

For background information on how to specify a classpath in the WebSphere Administrative Console, see Setting up WebSphere Application Server V3.02 for debugging servlets, EJB, and JSP, Setting up WebSphere Application Server V3.5 for debugging servlets, EJB, and JSP , and Setting up WebSphere Application Server V4.0 for debugging servlets, EJB, and JSP in the IBM Distributed Debugger documentation for the different versions of WebSphere Application Server.

Start the Object Level Trace server

In this section you start the OLT server and set its options so that you can debug your Web applications.

1. Open or switch to a Windows command prompt, or click **Start > Run**.
2. Start OLT by entering the command `olt`. If you want, you can create a shortcut to this command and place it on your desktop for future use.
3. On the **Client Controller** tab, select **Trace and debug** from the **Execution mode** drop-down list.
4. Type your Windows machine name (without the domain) in the **Debugger hostname** field.
5. Leave the default **Debugger TCP/IP port**. Click **Apply**.
6. Click **Options** and set **Step-by-step Debugging Mode** if it is not already set.

Object Level Trace is now enabled and is waiting for requests to trace client applications from any server on the network.

Debug the snoop servlet

The snoop servlet is a simple application that provides information about the servlet environment it is running in, for diagnostic or learning purposes. This application is provided as part of your WAS installation and has been compiled with debugging information so that you can debug it using OLT. Now that you have started the OLT server and set the debugging options in the WebSphere Administrative Console, you can start the servlet from a Web browser and then use OLT to launch a Distributed Debugger session.

Quick steps

1. Start the iSeries debug server job with the command `STRDBGSVR`
2. Open a Web browser and enter the URL: `http://`*`myhost`*`:`*`httpport`*`/servlet/snoop`
3. When the OLT Method breakpoints window opens, select `doGet:SnoopServlet...` method.
4. After a few seconds, the Distributed Debugger starts, and runs to the first debuggable line of code.
5. Click ↷ or ▶ to step over or run the method. The results page should appear in the Web browser.

Detailed Steps

1. Start the iSeries debug server job by issuing the command `STRDBGSVR` from a 5250 emulation window. You can issue this command safely even if the server is already running.
2. Open a Web browser and enter the URL: `http://`*`myhost`*`:`*`httpport`*`/servlet/snoop` where *myhost* is the iSeries system and *httpport* is the port for your HTTP instance.
3. OLT displays the Method breakpoints window when it receives the trace signal from your WAS instance (this may take up to a minute from the time you enter the Web address). Select the `doGet:SnoopServlet...` method and click **OK**.
4. Enter your iSeries system user ID and password if prompted.
5. After a few seconds, the Distributed Debugger starts, and runs to the first debuggable line of code in `SnoopServlet.doGet()`.
6. Click ↷ repeatedly to watch execution flow, then click ▶ to continue running the program. The results page should appear in the Web browser.

If the Web application completes and OLT did not intercept the call to the snoop servlet, or if the Debugger was unable to load the servlet, you may not have configured WAS or OLT correctly. Revisit “Set debugging options in WebSphere Administrative Console” on page 54 and “Start the Object Level Trace server” on page 54 and ensure that WAS and OLT are correctly configured. Make sure that any paths in your CLASSPATH in WAS are separated by colons (:) not semicolons (;). Finally, make sure that the CLASSPATH for the snoop servlet is part of the classpath specified in the command line arguments for the Default server.

Summary

There are many techniques for starting a debug session for Java-based Web applications on your iSeries system. You can use Object Level Trace to intercept method calls for servlets in your WAS applications, and decide which methods you want to debug. Or you can connect the Distributed Debugger to a WAS application job, load the appropriate Java classes, set breakpoints in appropriate methods, and run the program. Which method you choose depends on the type of program you are debugging and what your objectives are. For example:

- For a simple servlet application that always involves a call to doGet() or doPost() to handle requests, OLT gives you a quick way to start debugging calls to either method. These methods are usually the first methods invoked after you start the servlet from a Web browser.
- For a very complex Web application involving many servlets, JSPs, and EJBs, OLT helps you trace the calling relationships between program components, and is a good choice if you want to follow program flow and dynamically choose what methods to debug.
- For a complex Web application where you are only interested in debugging one method (or in determining whether a particular section of code is ever traversed), OLT may involve too many interruptions for method calls you do not want to debug; in this case, you can attach the debugger to the application server, load the Java class, set breakpoints, and run your application; if it runs to completion, your breakpoint was never reached.
- For a Web application that you are unable to start debugging from OLT, you can attach to the Application Server job in the Distributed Debugger, and then try to load the program to determine why you cannot debug it from OLT. If you cannot load the application in the debugger, you probably cannot run the application without debugging as well, and this provides an indication that something more serious than a program logic error is causing the failure. For example, the .class file may not be valid.

For more detailed information on the Distributed Debugger or Object Level Trace, see the appropriate entries in the **Help** perspective of the workbench.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

Director of Licensing
Intellectual Property & Licensing
International Business Machines Corporation
North Castle Drive, MD - NC119
Armonk, New York 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Canada Ltd. Laboratory
Information Development
B3/KB7/8200/MKM
8200 Warden Avenue
Markham, Ontario
Canada L6G 1C7

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Programming Interface Information

This publication is intended to help you to create and manage applications and user interfaces on the workstation, in a client/server environment. It contains examples of data and reports that are used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses that are used by an actual business enterprise is entirely coincidental. This publication documents General-Use Programming Interface and Associated Guidance Information provided by IBM WebSphere Development Tools for iSeries.

Trademarks and Service Marks

The following terms are trademarks or registered trademarks of the International Business Machines Corporation in the United States or other countries or both:

Application System/400
AS/400
AS/400e
DB2/400
IBM
iSeries
Integrated Language Environment
OS/400
VisualAge
WebSphere

Java and all Java-based trademarks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation in the United States, or other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Program Number: 5724-A81

Printed in U.S.A.

SC09-2625-06